DECEMBER 1975 $1.50

BYTE

the small systems journal

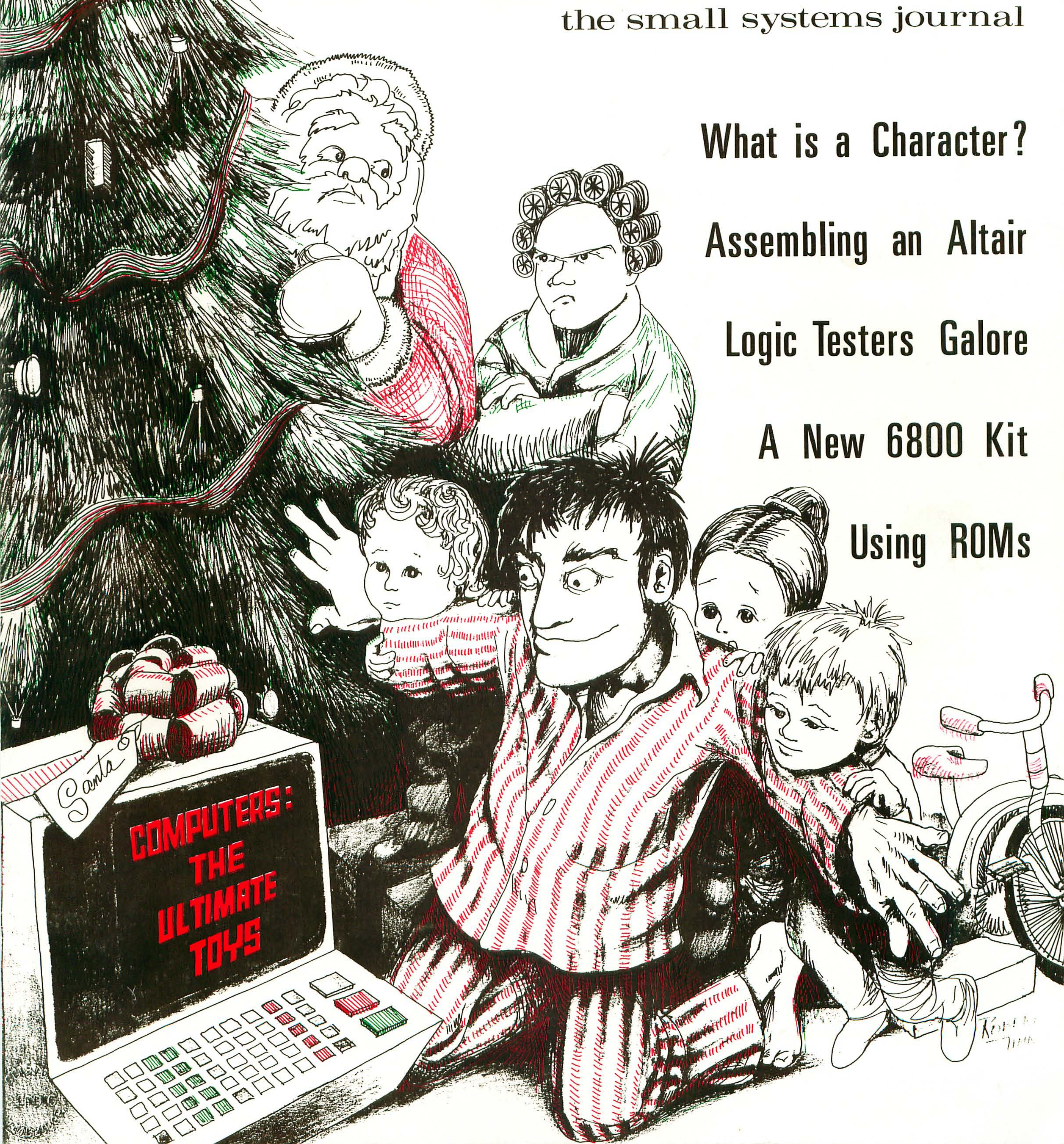
What is a Character?

Assembling an Altair

Logic Testers Galore

A New 6800 Kit

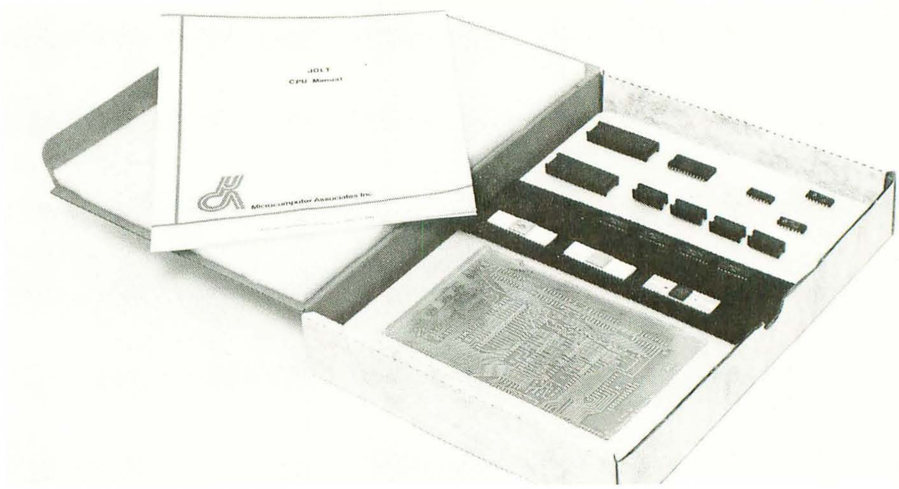
Using ROMs



Introducing ...the world's lowest cost computer system JOLT™

JOLT™ is the new, fully-tested microcomputer with the exclusive on-board DEMON™ debug monitor. You can build it, plug it in and talk to it in three hours or less . . . for a price of just $249!

The basic JOLT™ card includes an 8-bit MOS Technology Model 6502 CPU, which requires no clock, can directly address 65k of memory, has two index registers, 58 instructions with 11 addressing modes, two interrupts and includes both single step and address halt capability. And that's only a part of it.



All kits are delivered with a complete instruction manual and packaged for easy visual identification of parts to aid you in assembly.

JOLT's™ CPU card is available IMMEDIATELY* in either kit form or assembled ($249 for the kit in single quantity and $348 assembled). Either way, the JOLT™ CPU is completely tested prior to delivery. It comes complete with a terminal interface (TTY or EIA) and a unique software DEbugger/MONitor called DEMON™, for which full documentation is provided. It is very easy to program, and any JOLT™ delivery includes an easy-to-follow assembly instruction manual, showing you exactly how to put everything together . . . correctly. Complete assembly should take you no more than three hours if you choose the CPU in kit form. Besides the JOLT™ CPU — the 6502 from MOS Technology — the basic JOLT™ card has a fully static memory accommodating 512 bytes of the user RAM. The JOLT™ CPU memory also has 64 bytes of interrupt vector RAM. ROM Program memory on the basic card consists of 1k bytes of monitor/debugger with an automatic Power-On bootstrap program — so you can start talking to JOLT™ and it to you as soon as you plug it in to your terminal. On-board Input/Output devices on the JOLT™ CPU card include TTY 20 milliamp current loop and an EIA interface, both full duplex. The card has high speed reader interface lines and 16 fully programmable user I/O lines with full TTL drive.

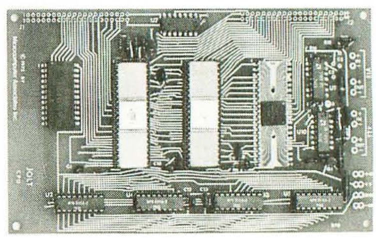
Nobody, but nobody, except MAI can offer you an on-board debugger/monitor like DEMON™. It's fully documented, too.

The exclusive DEMON™ Debug Monitor really makes JOLT™ one of the most outstanding computer systems offered at any time, at any price. Even without DEMON™ and its superior software features, JOLT™ is the lowest cost computer system in existence. And DEMON™ is a bonus you'll have to use to believe. First, it self-adapts to any terminal speed from 10-30 CPS. With it, you can display and alter your CPU register and memory locations, plus you can read, write and punch Hex formatted data . . . with Write/Punch BNPF format data for PROM programmers. It has unlimited breakpoint capability along with separate non-maskable interrupt entry and identification. External device interrupts can be directed to any location you choose, or they can be defaulted to DEMON™ recognition. DEMON™ also gives you (1) a completely protected ROM resident debug/monitor; (2) the capability to begin execution at any location in memory; (3) the capability to bypass DEMON™ entirely to permit full control by you over your system; (4) a high-speed 8-bit parallel input option; and (5) it includes user callable DEMON™ I/O subroutines. MOST IMPORTANT, DEMON™ IS INCLUDED AS STANDARD WITH ANY JOLT™ CPU KIT OR ASSEMBLED BOARD!

Obviously, the JOLT™ basic card is a computer in and of itself. But you can add significantly to its capacity and versatility by adding 4k RAM JOLT™ memories — in one card or a whole bunch. A RAM card kit is only $265 ($320 assembled). Now. *4096 Bit RAM 4K BYTE*

The JOLT™ memory card is a fully static 4,096-bit Random Access Memory (RAM) with 1 microsecond access time and on-board decoding. It is also available now.* And the quantity of one price is what you might expect to pay in quantities of 100 . . . very inexpensive!

There's also a JOLT™ I/O card for you, our Peripheral Interface Adapter. You can't beat the price — single kit 96 bucks — or the function.



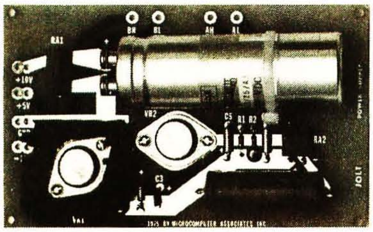
Pictured above is the assembled JOLT™ CPU card with DEMON™. just plug it in and you're ready to go.

The JOLT™ PIA (Peripheral Interface Adapter) I/O card includes two PIA LSI chips, 32 input/output lines, two interrupt lines, on-board decoding and standard TTL drive. It is also fully programmable and available IMMEDIATELY* in either kit or assembled form . . . at a very attractive single unit price ($140 assembled).

Considering the function and capacity of the JOLT™ Power Supply Card, you probably think the quantity of one price — $145 — is a misprint. It isn't.

The JOLT™ family also includes a power supply card, which operates at any of

three voltages — +5, +12 and −10. The power supply supports the basic JOLT™ CPU card, plus 4,096 bytes of RAM and I/O. The only two words for the price are "dirt cheap." It is available for delivery immediately with a single unit kit price of $145 ($190 assembled).



The assembled power supply card shown above powers the JOLT™ CPU, I/O, and RAM Memory cards.

You can also choose a blank JOLT™ universal card. Or several.

The JOLT™ Universal card is completely nude. It's a blank you can use any way you wish, for control panels, T.V. interfaces, keyboards, LED's, or any other interface logic, because the card's holes are drilled to accept 14, 16, 24, or 40 pin sockets and has the same form factor as the other JOLT™ cards. The single unit price is just $25.

If you think you need extra cables, wires and the like, choose a Super Value JOLT™ Accessory Bag. A $55 Value for just $40.

The JOLT™ Accessory Bag includes 25 separate parts, enough hardware to connect one JOLT™ card to another. Order an Accessory Bag for each additional JOLT™ card. The Bag contains such necessary items as flat cable, connectors, cord spacers, hardware, wire, etc.

THE JOLT PLAIN ENGLISH WARRANTY

All components in the JOLT™ family are new and fully tested prior to shipment. Kit components are fully warranted during the first 60 days of ownership. Assembled parts are fully warranted during the first 6 months of ownership. If your properly assembled kit does not work, just ship your order back to Microcomputer Associates Inc. and we'll repair, replace, or refund your money.

20% BONUS

NOW! Special ONE TIME BONUS DISCOUNT. Order one CPU before **November 10, 1975** and deduct 20% off of all additional cards and accessories. (Note: Discount does not apply to CPU cards, assembled or unassembled.)

We told you JOLT™ was the world's lowest priced computer system. These prices prove it.

JOLT KITS
(ALL PAYMENTS MUST BE IN U.S. DOLLARS)
QUANTITY PRICING

	1-4	5-19	20-49	50-99	100 up
JOLT™ CPU	$249	$235	$230	$225	$215
JOLT™ RAM	265	255	250	245	235
JOLT™ I/O	96	90	88	82	75
JOLT™ POWER SUPPLY	145	135	130	128	115
JOLT™ UNIVERSAL CARD	25	20	18	16	14
JOLT™ ACCESSORY BAG	40	36	34	32	30

ALL JOLT™ KITS ARE DELIVERED COMPLETE WITH A DETAILED AND ILLUSTRATED ASSEMBLY MANUAL

** Order will be shipped within 15 days.*

JOLT™ ASSEMBLED PRICES
(ALL PAYMENTS MUST BE IN U.S. DOLLARS)
QUANTITY PRICING

	1-4	5-19	20-49	50-99	100 up
JOLT™ CPU	$348	$325	$320	$315	$305
JOLT™ RAM	320	305	300	295	280
JOLT™ I/O	140	125	120	115	105
JOLT™ POWER SUPPLY	190	175	170	165	150

THE NEW JOLT™ FROM MAI IS AVAILABLE ONLY ON A DIRECT BASIS. USE THIS COUPON TODAY TO ORDER FROM

Microcomputer Associates Inc.
111 Main St., Department G
Los Altos, Calif. 94022
Phone (415) 941-1977

This coupon will bring you a JOLT™! Complete and return it.

Here is my check, money order or credit card payment for the following JOLT™ products. If a U.S. or Canadian order, please add $4.86 for shipping, handling and insurance. (Orders to all other countries are priced F.O.B. Los Altos and will be shipped FREIGHT COLLECT.)

Send me immediately:

QUANTITY	PRODUCT	PRICE
________	JOLT™ CPU KIT	$ ________
________	JOLT™ RAM MEMORY KIT	________
________	JOLT™ I/O KIT	________
________	JOLT™ POWER SUPPLY KIT	________
________	JOLT™ UNIVERSAL CARD	________
________	JOLT™ ACCESSORY BAG	________
________	JOLT™ CPU ASSEMBLED	________
________	JOLT™ RAM MEMORY ASSEMBLED	________
________	JOLT™ I/O ASSEMBLED	________
________	JOLT™ POWER SUPPLY ASSEMBLED	________
	TOTAL	$ ________
+U.S./CANADA SHIPPING/HANDLING/INSURANCE		4.86
NOTE: CALIFORNIA RESIDENTS, +6% SALES TAX		________
TOTAL REMITTANCE (U.S. DOLLARS ONLY)		**$** ________

MAIL YOUR ORDER TO: Pehaco Corporation, JOLT™ Sales Agents
Microcomputer Associates Inc. Dept. G
111 Main St., Los Altos, Ca 94022

All orders subject to the prior approval of the manufacturer.

20% BONUS NOW! Special ONE TIME BONUS DISCOUNT. Order one CPU before **November 10, 1975** and deduct 20% off of all additional cards and accessories. (Note: Discount does not apply to CPU cards, assembled or unassembled.) **20% BONUS**

PLEASE CHARGE THE TOTAL TO MY CREDIT CARD:
BANKAMERICARD # ____________________

MASTER CHARGE # ____________________ (include 4-digit #)

My credit card expires on ____________________

SIGNATURE ____________________ All credit card orders must be signed!)

I UNDERSTAND THAT ALL U.S. ORDERS WILL BE SHIPPED VIA U.P.S. AND ALL INTERNATIONAL ORDERS WILL BE SHIPPED BY AIR. SEND MY ORDER TO:

NAME ____________________

ADDRESS ____________________

ORGANIZATION ____________________

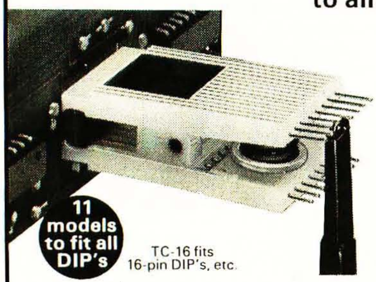
CITY ____________ STATE ____________ ZIP ____________

PHONE ____________________ (Include Area Code)

***Make your check payable to: Microcomputer Associates Inc.**
International Orders: Payment should be cabled to Account #00417-06355 Bank of America, 215 So. Murphy Avenue, Sunnyvale, California, USA, 94086

IC troubles? try these high-performance Super-Grip™ IC Test Clips

for fast, non-shorting access to all leads on dual-in-line IC packages



No more shorting across DIP leads... just quickly clip on an IC TEST CLIP to bring DIP leads out for safe attachment of scope probes and other leads. Ideal for signal input, tracing, troubleshooting, etc. Patented precision, "contact comb" design guarantees no shorting between DIP leads. Probes can hang "no-hands" free on Test Clip terminals in card racks (unique — see photo). Engineered Mechanical clamping plus gold-plated phosphor bronze terminals provide superior electrical contact. *Also unequaled as a DIP removal tool.*

Model	Row-To-Row Dim.	Part Number	Price
TC-8	.3 IN.	923695	$7.35
TC-14	.3 IN.	923698	4.50
TC-16	.3 IN.	923700	4.75
TC-16 LSI	.5/.6 IN.	923702	8.95
TC-18	.3 IN.	923703	10.00
TC-20	.3 IN.	923704	11.55
TC-22	.4 IN.	923705	11.55
TC-24	.5/.6 IN.	923714	13.85
TC-28	.5/.6 IN.	923718	15.25
TC-36	.5/.6 IN.	923720	19.95
TC-40	.5/.6 IN.	923722	21.00

We honor M.C. and B.A.C. charges.
Add sales tax on OH and CA orders.
(F.O.B. Painesville on company P.O.'s.)
Dealer inquiries invited

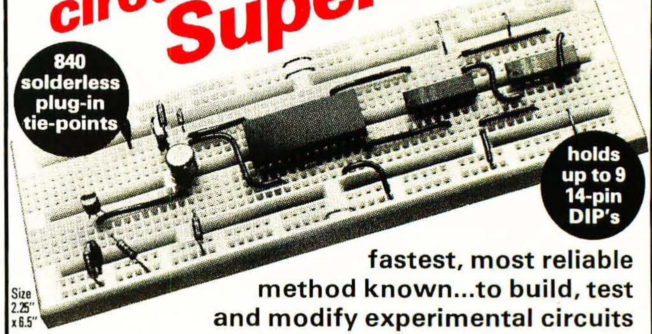
Add ⟶ fees from this chart.

SHIPPING/HANDLING		C.O.D.
Up to $10.00	$1.00	$ .70
$10.01 to $25.00	1.50	.80
25.01 to 50.00	2.00	.90

All products guaranteed to meet or exceed published specifications

A P PRODUCTS INCORPORATED
Box 110-G • Painesville, OH 44077 • 216/354-2101

Particular? try this high-performance circuit-builder's Super-strip™



fastest, most reliable method known...to build, test and modify experimental circuits

8-bus distribution system plus universal .1" by .1" matrix of solderless, plug-in tie points all in one integral device.

Accepts all DIP's and discretes with leads up to .032" dia.

Interconnect with any solid wire up to No. 20 A.W.G.

ORDER BY PART NUMBER
923252 (nickel-silver terminals)....... **$17.00**
923748 (gold-plated terminals)........ **$18.90**

High-performance A P Super-Strip component matrices consist of 128 terminals of 5 tie points each. The 8 buses of 5 connected 5-tie-point terminals are for voltage, ground, reset and clock lines, shift command, etc. New, non-shorting, instant-mount backing and quick-removal screws are supplied... use several Super-Strips to create large-scale breadboards on panels up to 1/8" thick.

We honor M.C. and B.A.C. charges.
Add sales tax on OH and CA orders.
(F.O.B. Painesville on company P.O.'s.)
Dealer inquiries invited

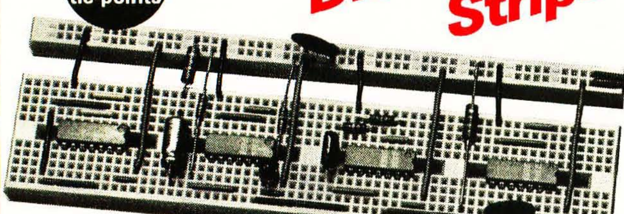
Add ⟶ fees from this chart.

SHIPPING/HANDLING		C.O.D.
Up to $10.00	$1.00	$ .70
$10.01 to $25.00	1.50	.80
25.01 to 50.00	2.00	.90

All products guaranteed to meet or exceed published specifications

A P PRODUCTS INCORPORATED
Box 110-G • Painesville, OH 44077 • 216/354-2101

Creative? try these high-performance circuit-builder's Terminal and Distribution Strips



Super-Versatile™ building blocks for experimental circuits

Universal .10" matrices of solderless, plug-in tie points

For all DIP's and discretes with leads to .032" dia.

Interconnect with any solid wire up to No. 20 A.W.G.

Nickel-silver terminals

ORDER BY PART NUMBER
923277 distribution strip . . . **$2.50**
923261 terminal strip **$12.50**

Create custom breadboards in minutes with these new instant-mount strips. DISTRIBUTION STRIP (top) contains 2 continuous buses of 12 connected 4-tie-point terminals. Size: 6.5" by .35". TERMINAL STRIP contains 128 5-tie-point terminals, holds up to *nine* 14-pin DIP's. Size: 6.5" by 1.36". Integral, non-shorting, instant-mounting backing permits quick build-up of special breadboards using any mix of strips.

Other models available.

We honor M.C. and B.A.C. charges.
Add sales tax on OH and CA orders.
(F.O.B. Painesville on company P.O.'s.)
Dealer inquiries invited

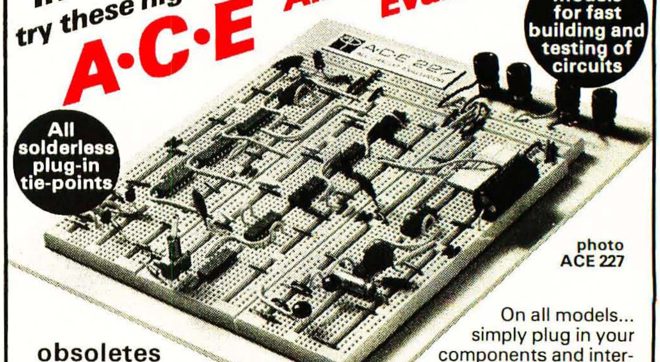
Add ⟶ fees from this chart.

SHIPPING/HANDLING		C.O.D.
Up to $10.00	$1.00	$ .70
$10.01 to $25.00	1.50	.80
25.01 to 50.00	2.00	.90

All products guaranteed to meet or exceed published specifications

A P PRODUCTS INCORPORATED
Box 110-G • Painesville, OH 44077 • 216/354-2101

In a hurry? try these high-performance A·C·E All-Circuit Evaluators



obsoletes ordinary breadboards

On all models... simply plug in your components and interconnect with 22-ga. solid wire. All models accept all DIP's, TO-5's and discretes with leads up to .032" diameter.

Multiple buses can easily be linked for power and ground distribution, reset and clock lines, shift command, etc. Bases: gold-anodized aluminum. Terminals: non-corrosive nickel-silver. Four rubber feet included.

Order No.	ACE Model No.	Tie Points	DIP Capacity	No. Buses	No. Posts	Board Size (inches)	Price Each
923333	200-K (kit)	728	8 (16's)	2	2	4-9/16 x 5-9/16	$18.95
923332	208 (assem.)	872	8 (16's)	8	2	4-9/16 x 5-9/16	28.95
923334	201-K (kit)	1032	12 (14's)	2	2	4-9/16 x 7	24.95
923331	212 (assem.)	1224	12 (14's)	8	2	4-9/16 x 7	34.95
923326	218 (assem.)	1760	18 (14's)	10	2	6-1/2 x 7-1/8	46.95
923325	227 (assem.)	2712	27 (14's)	28	4	8 x 9-1/4	59.95
923324	236 (assem.)	3648	36 (14's)	36	4	10-1/4 x 9-1/4	79.95

We honor M.C. and B.A.C. charges.
Add sales tax on OH and CA orders.
(F.O.B. Painesville on company P.O.'s.)
Dealer inquiries invited

Add ⟶ fees from this chart.

SHIPPING/HANDLING		C.O.D.
Up to $10.00	$1.00	$ .70
$10.01 to $25.00	1.50	.80
25.01 to 50.00	2.00	.90

All products guaranteed to meet or exceed published specifications

A P PRODUCTS INCORPORATED
Box 110-G • Painesville, OH 44077 • 216/354-2101

In the Queue

Foreground

POWERLESS IC TEST CLIP 26
Test Equipment – Baker – Errico
LIFE Line 3 48
Applications – Helmers
BUILD A 6800 SYSTEM WITH THIS KIT 72
Hardware Review – Kay
CAN YOUR COMPUTER TELL TIME? 82
Applications – Hogenson
PHOTOGRAPHIC NOTES ON PROTOTYPE CONSTRUCTION 94
Hardware – Helmers

Background

THE SOFTWARE VACUUM 12
Opinion – Ryland
LOGIC PROBES – HARDWARE BUG CHASERS 20
Tools – Burr
WHAT IS A CHARACTER? 30
Fundamentals – Peshka
FLIP FLOPS EXPOSED 58
Hardware – Browning
READ ONLY MEMORY TECHNOLOGY 64
Hardware – Lancaster
THE HP-65: WORLD'S SMALLEST COMPUTER 70
Systems – Nelson
ASSEMBLING AN ALTAIR 8800 78
Hardware – Zarrella

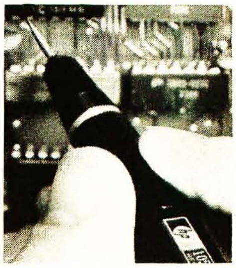
Nucleus

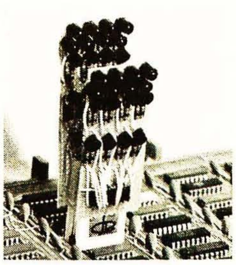
In This BYTE 4
What This Country Needs 5
BOMB 9
Diagnostics 10
BYTE's Bits 17, 88, 98
Word Hunt 18
Clubs, Newsletters 100
Letters 102
Book Reviews 108
The BYTE Questionnaire 112
Reader's Service 112

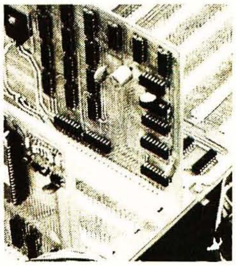
BYTE magazine is published monthly by Green Publishing, Inc., Peterborough, New Hampshire 03458. Subscription rates are $12 for one year worldwide. Two years, $22. Three years, $30. Second class postage application pending at Peterborough, New Hampshire 03458 and at additional mailing offices. Phone: 603-924-3873.

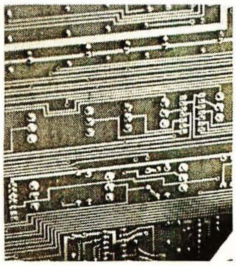
Address editorial correspondence to Editor, BYTE, Peterborough NH 03458.

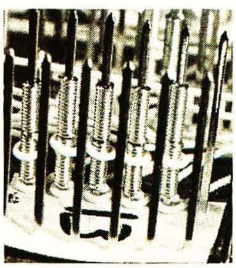
BYTE #4
DECEMBER 1975


p. 20


p. 26


p. 72


p. 78


p. 94



In This BYTE

In the Christmas BYTE you'll find the following morsels:

To quote an ancient philospher, "nature abhors a vacuum." Chris Ryland's **Opinion: The Software Vacuum** describes a void in the personal use computer marketplace. Will nature in the form of profit motive come in and fill the software vacuum? Only time will tell . . .

Strange things sometimes occur in the electronic pathways of a computer. Putting on your detective hat may occasionally be required – in which case Alex. F. Burr's review **Logic Probes – Hardware Bug Chasers** will give you valuable information on several commercial products which can help debug your designs.

On the same theme but in the foreground this time, Robert Baker and John Errico have provided an article on a fairly sophisticated **Powerless IC Test Clip** which you can construct for $20 or so in parts. For the do-it-yourselfer, this design results in 16 little binary voltmeters which can be clamped onto an integrated circuit to examine the logic levels at each pin.

What is a Character? You can find out by reading Manfred Peshka's tutorial on some of the basic concepts of programming and information systems work. Old hands at the programming arts will find this to be an interesting review, and readers new to programming will find it necessary background material.

After an interruption, the LIFE Line series continues this month with the third installment. In **LIFE Line 3** you'll find the beginning of information on the interactive commands which are decoded by the program.

The Flip Flop is an important element in designs used with computer chips and peripherals. William E. Browning has provided this article to introduce the less experienced readers to this fundamental building block.

Read Only Memory Technology can be used in situations ranging from clever logic and interface design to storage of systems programs in a computer. In his article on the subject in this issue, Don Lancaster gives some background information about ROM applications and several suggestions concerning their use as design elements.

What is **The World's Smallest Computer System?** Well, at this time it looks like the HP-65 pocket-size programmable calculator might qualify for the title. Find out why by turning to Richard Nelson's article on the subject.

The last BYTE featured a comparison of the Motorola 6800 CPU chip with a new contender from MOS Technology. In this issue, Gary Kay of Southwest Technical Products Corporation presents some information on the Motorola 6800 package his firm is supplying. What SWTPC has done is to take the standard parts, combine them with an attractive case, power supply and PC boards – and put the result into a package as a kit for readers to build.

What is it like to build an Altair computer kit? In his **First Person Report: Assembling an Altair,** John Zarrella describes his experience with the MITS product, from his decision to purchase, through assembly and hardware debugging.

Computers are fundamentally synchronous machines – they beat to the tune of a periodic clock. With program timing loops, a computer can be made to count the beats of its clock. You can find out how to do this by reading Jim Hogenson's article **Can Your Computer Tell Time?**

When assembling complicated logic systems, one of the best methods for new and experimental work is use of solderless wire wrap interconnection. Some pointers on prototype assembly are found in **Photographic Notes on Prototype Construction.**

And on the cover, artist Robert Tinney illustrates the impact of these new toys upon traditional relationships.

What This Country Needs Is a Good 8-Bit High Level Language

Editorial by Carl Helmers

Have you ever tried to talk to a person who speaks a language other than your own? You know that the person must think as a rational, sentient being or you wouldn't make the attempt to communicate. As a *human being* your conversational partner's basic thought patterns are of necessity similar. Yet you can't understand him and he can't understand you. There are ways around this problem, given a sincere interest in communications by both parties.

An analogous problem exists in the field of personal computing as it is practiced by the readers of BYTE. If you translate the words referencing human beings into words referencing computers and reread the preceding paragraph, the result will be a corresponding statement about the computer communication problem: In the home brew computer world, there are a number of different computer architectures, all speaking different machine languages. There are even dialects of machine languages since each home brew designer or manufacturer makes specific choices about address space allocations and input/output port assignments used with the standard computer chip designs. Yet all of these machines are potentially programmable to do similar functions.

Like two human beings facing each other but speaking different languages, computers can talk to each other on compatible channels. But making sense – that is to say, executing the same programs – is another matter. It is possible to connect an Altair up to a Scelbi 8H product using an RS-232 interface. The link can be made and every bit of every byte sent (for example) from the Altair will be received and digested by the Scelbi. But unless the machines have common referents and means of translation, the information sent will be no better than "noise" – just as a foreign language perceived by a person conveys no more meaning than an arbitrary sound until the language is learned. Having a working communications channel does not guarantee that the communications sent will mean anything.

Levels of Intelligence

Given two or more different computers and compatible electrical interfaces between them, the simplest and easiest form of common understanding is at the level of raw data to be processed by the different computers. A binary number is a binary number independent of the machine for which it was generated. The fact that the bit string 11000111 means "load accumulator from memory" in the machine language of an 8008 and means "You fool – that's an unimplemented op code" for the Motorola 6800 is an accident of hardware design at two different companies. In either case, as data, the binary number 11000111 means an integer value of 199. Similarly, an ASCII character is an ASCII character independent of the place and time of its creation. Given the decision to use ASCII for communications – or binary numbers, for that matter – both parties to the communication can talk characters or numbers. These simple data formats provide an easily implemented link between computers which

BILL GODBOUT ELECTRONICS & CAVE grafiX™

WISH YOU A JOYOUS HOLIDAY SEASON AND HOPE THAT '76 WILL BE YOUR BEST YEAR EVER

If you are into toyful tinkering as well as heavy computer stuff - you might like to check out...

BIGGER DIGITS... give your eyes a break with our 3/10" displays

WE HAVE A SUCCESSOR TO OUR VERY POPULAR "CHEAP CLOCK KIT"!

BRIGHTER DIGITS... segment and driver transistors included

SON -OF- A CHEAP CLOCK

MORE RELIABLE DIGITS... sockets included for IC and for readouts

THE RIGHT NUMBER OF DIGITS... 6 digits for hours, minutes, and seconds. Use as 12 or 24 hour clock, 50/60 Hz

AND NOT JUST THE DIGITS... We include all parts you need to make a working clock, including the PC board, transformer, all components. You supply the case and hardware.

$14.50

"Electronic Projects for Musicians"

BY CRAIG ANDERTON $6.95 +shpg

AN IDEAL BOOK FOR PEOPLE INTO MUSICAL ELECTRONICS...contains 4 chapters for beginners on practical electronics, so that even complete novices can create working devices. The book then describes 19 projects suitable for <u>any</u> level of experience, such as a ring modulator, preamp, mixer, battery eliminator, miniamp, bass fuzz, compressor, & 12 more. Also has sections on troubleshooting, finding more information, a forward by guitarist Joe Walsh, and a recording that demonstrates the sound of the projects.

they also make great gifts for any other electronics enthusiast you know. Scope our FLYER for more toys!

EROM BOARD SPECIAL OFFER

4K X 8 PRE-PROGRAMMED FOR 8080 $225

By special arrangement with a software house (meaning we pay them royalties), we've got 4K by 8 EROM boards pre-programmed with 8080 assembler, editor, and executive routines. It's still the same price as one of our stock 4K X 8 EROM board kits; we save enough on the programming (by doing lots of EROMs at a time) to absorb the software royalty so that you don't have to. And since these boards are EROM rather than mask, you can easily make changes in the program if desired. Speaking of changes: although these routines work out really well, we recognize the possibility that some bugs could pop up, or we may discover improvements as time goes on. That is why we're including 2 updates (at no extra cost) to purchasers of our board, should corrections or patches develop. Any changes you desire beyond that are available at a very nominal charge. We use our 8K x 8 EROM board as the basis---so it's expandable!

Toys! ★ BIGGER & BETTER ★ Toys!

PACE 16 bit chip set $195

1•PACE 16 BIT CPU 4•DS3608 1•DS0026
1•DM8837 32•2102 (4K BYTES OF RAM)
2•MM5204 (1K BYTE OF ROM) 1•PACE DATA PACKET

THIS CHIP SET GETS YOU INTO PACE THE EASY WAY --- INCLUDES ALL INTERFACE CHIPS FOR PACE TO SPEAK TTL, AS WELL AS RANDOM ACCESS MEMORY AND 1K BYTE OF EROM, AT A MONEY-SAVING PRICE. IF YOU DON'T NEED MEMORY, YOU CAN GET PACE & INTERFACE ICS FOR $125. IN EITHER CASE, WE GIVE YOU A PACE DATA PACKET SO YOU CAN START RIGHT IN!

PACE DATA PACKET $2.50

SINCE WE STARTED ADVERTISING AND SHIPPING PACE MICROPROCESSORS, WE'VE HAD A LOT OF RESPONSES AND QUESTIONS. IN FACT, WE'VE GOTTEN SO MANY QUESTIONS THAT IF WE SPENT ALL OUR TIME ANSWERING THEM ON A ONE-TO-ONE BASIS, WE WOULDN'T HAVE ANY TIME LEFT TO TAKE CARE OF BUSINESS. SO, WE GENERATED A COMPLETE PACE DATA PACKET---- MORE THAN 80 PAGES OF DETAILED AND SPECIFIC INFORMATION ON THE CHIP ITSELF, SOFTWARE, SYSTEM ORGANIZATION, AND MORE. ALSO INCLUDES AN 11 BY 17 INCH FOLDOUT LOGIC PRINT OF HOW TO MAKE THE PACE INTO A CPU BOARD THAT SPEAKS TTL. THE COST IS $2.50, REFUNDABLE WITH A PACE ORDER, TO COVER THE COST OF PRINTING AND SHIPPING.

4K X 8 RAM KIT ➡ ⇨ ➡ ⇨ ➡ $109.22

Note the lower price; we didn't think you'd mind. THIS KIT IS DIRECTLY **PLUG-IN COMPATIBLE** WITH THE ALTAIR 8800: NO JUMPERS, NO RASSLE. Also electrically compatible with other 8 bit machines.
With this kit you get sockets for all ICs, industrial-quality plated-through board, lots of bypassing, five voltage regulators to share the power load (less thermal problems, more reliability), typical 500 ns access time @ 25°C, and buffered addresses and outputs (no input presents more than 1 low power TTL load; output can drive 20 standard TTL loads). Requires 5V @ 1A at 25°C. Comes with assembly hints and logic print of the RAM board. Like Bill says, "it's the whole ball of wax".

EROM BOARD KITS

8K X 8 $352 (INTRODUCTORY PRICE) ADD $48 AND WE'LL PROGRAM IT FOR YOU

4K X 8 $200 ADD $25 AND WE'LL PROGRAM IT FOR YOU

2K X 8 $125 ADD $15 AND WE'LL PROGRAM IT FOR YOU

NOW YOU CAN STUFF UP TO 8K WORTH OF YOUR FAVORITE SOFTWARE ON A BOARD; OR, HAVE US DO THE PROGRAMMING FOR YOU. LIKE THE RAM KIT MENTIONED ABOVE, THIS BOARD IS **DIRECTLY** PLUG-IN ALTAIR 8800 COMPATIBLE. INCLUDES: SOCKETS, BYPASSING, INDUSTRIAL-QUALITY PLATED-THROUGH BOARD, BUFFERED ADDRESSES AND OUTPUTS, ON BOARD REGULATION. LOW POWER CONSUMPTION; FULL 8K REQUIRES ONE-HALF AMP @ 5V AND 150 MA @ -12V. FILE PROTECT FEATURE INCLUDES LINE RECEIVER WITH REAL HYSTERESIS TO PREVENT FALSE TRIGGERS DUE TO NOISE. **EXPANDABLE**---IF YOU BUY OUR 2K BOARD AND WANT TO MOVE ON TO BIGGER AND BETTER THINGS (LIKE OUR 8K BY 8), ALL YOU NEED ARE SOME MORE SOCKETS AND EROMS. YOUR BOARD DOESN'T BECOME OBSOLETE AS YOUR SYSTEM EXPANDS.

THERE'S MORE...WE HAVE

READOUTS, LEDS, TRANSISTORS, FETS, LINEAR ICS, TTL, CMOS, MEMORIES, HARD-TO-FIND INTERFACE CHIPS, RESISTORS, CAPACITORS (ELECTROLYTIC--MYLAR--POLYSTYRENE--DISC), THE HOBBYWRAP TOOL, THE VECTOR WIRE PENCIL, VECTORBOARD AND PC BOARD, DIODES, INDUCTORS, A WHOLE BATCH OF POWER SUPPLY KITS, A COUPLE CLOCK KITS, OUR 12 VOLT 8 AMP HEAVYWEIGHT BENCH SUPPLY, MUSICIAN/AUDIO KITS, DIP SWITCHES, RIBBON CABLE, SPECTRA-STRIP, ROM PROGRAMMING SERVICE, ALL DIFFERENT KINDS OF SOCKETS...**SEND FOR OUR NEW WINTER FLYER AND GET THE FULL STORY.**

GODBOUT
BILL GODBOUT ELECTRONICS
BOX 2355, OAKLAND AIRPORT, CA 94614

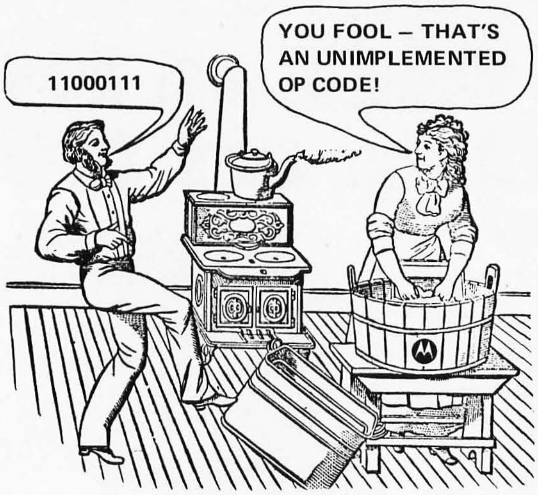
YOU MAY PLACE MASTERCHARGE® OR BANKAMERICARD® ORDERS BY CALLING, 24 HOURS A DAY,
(415) 357-7007
TERMS: ADD 50¢ TO ORDERS UNDER $10. ADD SHIPPING WHERE INDICATED; OTHERWISE, ITEMS SHIPPED POSTPAID. CAL RES ADD TAX. NO COD---IT'S TOO MUCH PAPERWORK!

does not require a large amount of software intelligence on the part of receiving and sending computers. It does not matter whether such a link is in the form of tape cassettes sent in the mail or a direct RS-232 connection when all the neighborhood hackers get together for a multiprocessor powwow and computerized crap game. ASCII and binary numbers can be shuffled back and forth with the assurance that, at this data level of coding, the information will be understood by both parties.

The communication represented by binary numbers or ASCII encoded data is not at all what might be called *true understanding* between the two computers involved. Sending data is a first step, but it is by no means the ultimate. The significance of the communication at this level must be determined by the *human beings* who manipulate the data being sent. There is no direct way of affecting the understanding — the programming — of the computer which is dutifully receiving the ASCII or binary data in this simple fashion.

Simple transmission of data across a parallel-serial-parallel or parallel-parallel interface enables the *users of the computers* to talk to one another, but the computers which carry out the exchange "couldn't care less." The computer in this type of an exchange is simply serving as a "dumb" transmission channel.

BASIC is an adequate language — but it has its drawbacks.



Borrowing again from the analogy to human beings, this *data level* communication between computers might be compared to the non-verbal emotional forms of interpersonal exchange. Common languages and verbal understanding are not required for humans to exchange emotional states, using music, facial gestures and other body motions which are inherent in the nature of the beast. But to exchange knowledge and practical data humans must speak a common language; it is not an accident that verbal activity and the tools of language are so much a part of the dominant species on this planet. If merely sending data represents a low level of communications intelligence between computers, what does it mean to transmit a higher level of intelligence?

Program Level Intelligence

To transmit data between computers is the first step toward a higher level of communications among diverse types of computers. Every computer on the market can handle 7-bit ASCII and other forms of information encoded into 8-bit bytes. (Of course, it may be easier to program ASCII data manipulations on some machines than on others.) Having a computer which can easily be *programmed* (possibly with no human intervention) as a result of an ASCII exchange with a computer of a different internal architecture is the essence of this next level of intelligence in intercomputer communications. In such *program level* exchanges the computers are speaking to each other in the form of *abstract program representations* which can be automatically translated into specific machine language representations for execution.

In the previous analogy, this is like human beings exchanging information and thought in a commonly understood language — for example, English. Each person who understands the language has incorporated within his mind a "translator" which creates an internal understanding based upon what was heard. The result of this translation — which must be consciously performed — is an understanding of the message which can be used as a basis for further thoughts. Emotional data is a much more directly perceived input (although it may of course be colored by thoughts). But verbal inputs require cogitation and analysis before they can be used and integrated.

The Goal: Exchange of Programs

The goal of *program level* interchange between computers is thus the ability to communicate understandable and potentially executable *programs* between computers of different design. When this goal is achieved it will be possible for a reader in one corner of the technological world — for instance an 8080 user — to develop a neat little utility program which can be sent to a friend in another corner of the technological world who has just completed a different processor such as a PACE machine. Since the program is recorded and communicated using the *program level* techniques, the recipient need only read the ASCII representation from the communications channel and process this data with a suitable "translator" in order to obtain a new executable program for a different machine design.

This program level of exchange is a well known technique which has been developed very thoroughly over the past 15 years after computer science left its formative years of the 1950's. It is the technique of *high level languages* and *compilers.* The *language* is the machine independent *notation* for the programs which are to be exchanged. The *compiler* is the computer program which carries out the translation. (Variations on this technique of course exist; for instance some languages like BASIC and FOCAL rarely have compilers, but typically use "interpreters" instead to compile, then execute statements one by one.) In the computer world at large, of course, there is no unanimity about choices of languages — and there no

doubt will be considerable variation in program representation philosophies in this personal microcomputer field. Be that as it may, exchanges at the program level are needed and computer languages/compilers are the technique for accomplishing such exchanges with *minimum* machine dependence.

So that's the story behind the need for a good 8-bit high order language. The home brew computing field is much more extensive than the confines of just one computer architecture, and the technological problem of passing 8-bit bytes all over the place is not at all impossible. The need is there, but can it be satisfied?

What Exists Now?

What currently exists in the way of high level languages for the field of home brew computers is limited. Currently available is only one language — BASIC — which to a certain extent satisfies the need for a *good* language. BASIC now exists for the MITS Altair, and will soon be offered by several other manufacturers. As such it is the only high level language which both exists and will (hopefully) run the same programs on any one of these small computer systems. As a high level language, BASIC is adequate but it has a few drawbacks:

— Descriptive names of variables are impossible with single character identifiers.

— Only a primitive GOSUB/RETURN facility exists for subroutine linkage, and parameter passing is not built into the language.

— The language BASIC is interpretatively executed, which means that each statement is "compiled on the fly" and executed whenever it is encountered. (Pre-scanning of programs and reduction of the source text is sometimes done, however.) An interpreter is necessarily slower than an equivalent compiler's object code.

— BASIC is missing the more advanced software tools such as the IF-THEN-ELSE construct, and statement grouping constructs, like the PL/1 DO . . . END block.

— Only line numbers may be used to label places in the program.

Now don't make the mistake of concluding from this criticism that BASIC is useless. Far from it. Any high level language which works as well as BASIC is better than none at all in the majority of programming circumstances. This is because for most problems the minute details of execution are unimportant, provided that certain functional building blocks of software (provided by the higher order language) are available to use.

The BASIC language has been used as a tool for initially teaching computer programming concepts, and has done so from its inception in the early 1960's at Dartmouth. There are also innumerable tutorial books about BASIC, due to its widespread use in the educational field. It is certainly the case that in most implementations BASIC is a quick and conversational way to write simple programs at a terminal. The criticisms have to do with features in contemporary computer language technology which are not present in BASIC, but which are extremely useful when writing programs.

An Alternative to BASIC

Criticism without giving an alternative is an empty activity. The purpose of criticism is to find a way to a better approach. So what language exists, can be dreamed up, or adapted to the small systems context — and provides a better alternative to BASIC? At present there is one language which was expressly designed for systems programming and applications programming for microprocessors. This language is called PL/M, which is a registered trademark of the Intel Corporation. The origins of PL/M can be traced back to a book published in 1970 by three compiler specialists, W. M. McKeeman, J. J. Horning and D. B. Wortman called *A Compiler Generator* (Prentice-Hall, Englewood Cliffs NJ).

XPL is a subset of the IBM language PL/1. The XPL subset is designed to eliminate many extraneous bells and whistles from PL/1, retaining only those features most needed for writing compiler programs: Simple character string and binary data, manipulations of such data, and a block structured procedure oriented language. Another design criterion of XPL is that it had to be a simple language so that its compilers could easily generate efficient object programs without burning up incredible amounts of computer time. The authors of the language and the book describing it succeeded well, producing a powerful language design system which has been used in a number of large projects.

Now, as it turns out, the features which are in XPL are in many respects the features

In most programming circumstances, any high level language is better than no high level language at all.

PL/M is becoming an industry standard language: The computer language equivalent of a "black box" integrated circuit. BYTE would like to see a PL/M compiler adapted to the home brew computer context.

which are desirable for a programming language used with microcomputers for both applications and systems programming. XPL is not too far removed from assembly language and becomes very handy as a way to generate large programs without getting bogged down in details. This fact makes XPL a language of far more utility than a mere compiler writing tool.

When the time came for Intel to commission a high order language for programming of their microcomputers, the XPL language and compiler had been proven in several years of practical use by several compiler writing organizations. Its practicality as a systems programming tool no doubt resulted in the use of XPL as a model for the new PL/M language. PL/M is effectively an adaptation of XPL to the context of a microcomputer with 8-bit data quanta and 16-bit addressing. The result is a language which looks very much like XPL – with a few keyword substitutions and additional features. This resemblance is sufficiently close that at least one version of PL/M has been implemented simply by modifying a working XPL compiler, although Intel's original was implemented in FORTRAN.

PL/M as a language possesses many desirable attributes which are not found in BASIC. These attributes include the PL/1-like statements and statement groups, long descriptive names for variables and labels, block structure, and subroutine linkages with parameters. As a systems programming language for microcomputers, the PL/M language adopts some of the features of an absolute assembler – there are location counters for program code and data which can be set during a compilation. To top it all off, the PL/M language is a relatively simple one which can potentially be self-compiled upon a small (but not minimal) home brew system.

At the time of this writing, PL/M is fast on its way to becoming an industry standard. It is definitely a language which has the potential for adaptation to the software requirements of the more advanced programmers in BYTE's readership – yet at the same time it is simple enough for the novice to understand. At the present time, however, only *cross compilers* – large programs running on big machines – are available for PL/M. There are PL/M versions currently in the works or producing code for the 8080, the 6800 and PACE microcomputers – but all are cross compilers. These cross compiler versions are widely used via time sharing networks by a variety of industrial and commercial users of microprocessors. This acceptance indicates that PL/M is a language which is likely to be around for some time.

A Call For Compilers

So PL/M is the tool which the industrial and commercial world uses for efficient code generation with a high level language for microprocessors. Will this technology be made available in the home brew computer markets? Yes. One reason for writing this editorial is to point out the existence of PL/M and direct a few BYTE readers to appropriate sources of information. In future issues, BYTE will be getting into more of the details of PL/M as a language. Until then, the accompanying list of information sources will have to suffice.

A second reason for this editorial is to serve as a *call for compilers.* What is needed is a compiler for PL/M *or a similar language* which will run on a typical 16K (RAM) 8-bit microcomputer using as many as three serial I/O devices for multiple passes through the data of a source program. Ultimately there should be one such self-compiler program for each of the major microcomputer chip architectures. The compilers should be written with system design flexibility in mind (in other words, modularity throughout and isolation of hardware-dependent portions to specific modules). Who will be the first person, club or firm to provide such a self-compiler? ■

Information Sources

PL/M:
8008 and 8080 PL/M Programming Manual, MCS-451-0275-10K
Intel Corporation
3065 Bowers Ave., Santa Clara CA 95051
This describes 8008/8080 PL/M as originated by Intel.

PL/M6800:
PL/M6800 Programmers Reference and *PL/M6800 Language Specification*
Intermetrics, Inc.
701 Concord Ave.
Cambridge MA 02138
These manuals describe the version of PL/M being marketed for the 6800 processor.

As this issue goes to press, National Semiconductor has announced a version of PL/M called "PL/M+" for the PACE system. Further details will be provided by BYTE as they become available.

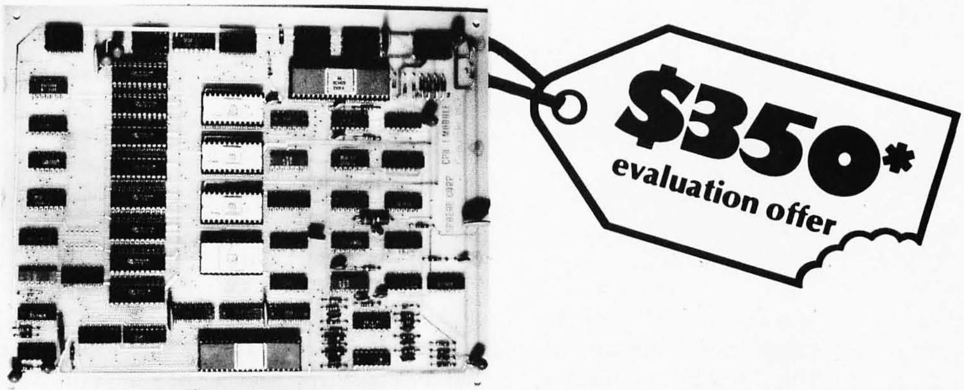
DIAGNOSTICS: Documentation of bugs in previous BYTEs.

BYTE #2, p. 54, Fig. 3. An inverter (e.g., 1/6 7404, or 1/4 7400) should be inserted between the CE inputs of the 7489 circuits and the 7400 which drives them in the original drawing. Thanks to Martin E. Haring, Edison NJ and several other readers for pointing this out.

Dan Clarke (105 Fir Court, Fredericton NB, Canada E3A 2E9) notes that the originate modem transmit frequency definitions (Fig. 14 and text of "Serial Interface"), page 35, BYTE #1, are incorrect. Using the Motorola *M6800 Microprocessor Applications Manual* page 3-32 as a source of data, the following table should correct the matter:

Mode	Data	Transmit Freq.	Receive Freq.
Originate	Mark	1270 Hz	2225 Hz
Originate	Space	1070 Hz	2025 Hz
Answer	Mark	2225 Hz	1270 Hz
Answer	Space	2025 Hz	1070 Hz

One-Card Computer.



- real-time clock
- 16 digital I/O lines
- 4k RAM
- 512 times 8 PROM
- serial teletype interface
- hardwired ROM monitor (console emulator)

(Complete with usage, programming, and application manuals.)

Still, our price goes down a lot easier.

Now's your chance to bite into a complete computer system. We at Sphere have used the latest microprocessing technology along with some real innovations in mini-circuit design to develop the lowest-cost complete computer systems available. We've found that most people find our CPU module hard to believe. This CPU can monitor functions on a real-time basis, analyze, and quickly arrive at answers in a desired format. It's expandable; and SPHERE offers a complete line of the lowest-cost peripherals on the market today.

KIT*	ASM*	
$350	$520	**ONE-CARD COMPUTER:** Includes Motorola 6800 microprocessor, 4K RAM, 512 bytes EPROM, (containing a Program Development System), a REAL-TIME CLOCK, 16 LINES OF DIGITAL I/O, serial type interface and hard wired ROM monitor.

*This is the OEM 100-quantity price, extended to the hobbyist, for a limited time, in quantities of one.

SPHERE CORPORATION
Dept. 1212
791 South 500 West #
Bountiful, Utah 84010

Dear SPHERE:

Yes! The facts I've read have convinced me!

☐ Enclosed is a check for the exact amount. Please send my ONE-CARD COMPUTER, for 60 day delivery.

☐ Please rush me more details on your low-cost computer system and peripherals. (Specific questions, if any, are attached.)

Print Name: ______________

Address: ______________

City and State: ______________

Zip Code ________ Phone: ________

SPHERE
CORPORATION
791 South 500 West Dept. 1212
Bountiful, Utah 84010
(801) 292-8466

Opinion:

The Software Vacuum

. . . I'd like to browse through several software catalogs . . . picking out the best package for my needs.

by
Chris Ryland
25 Follen St.
Cambridge MA 02138

There is a software vacuum. That fact has become increasingly clear in the past few months.

Take a look at the situation. At the time this is written there's only one company (MITS) offering a proprietary software product (BASIC) aimed directly at the microcomputer hobbyists. It's true that the People's Computer Company (PCC), the MITS Altair User's Group, and a handful of other user groups offer access to growing libraries of applications programs. Some microsystem manufacturers also supply "bare bones" system software with their machines, but the user is lucky if these bones consist of as much as a rather bare monitor, editor and assembler. Such software is just as important to a serious hobbyist's system as the hardware. So when I say there is a software vacuum, I mean there is an absence of commercially developed and marketed larger scale software products.

But why should this matter to me, as a hobbyist? What do I mean by larger-scale, and why should the hobbyist market be invaded by commercial software vendors? In answer, let me give a concrete example. Suppose I want to write a "desk calculator" program for my system. I'd like it to include all the scientific functions of a powerful calculator, such as sine, cosine, tangent, arcsine, the hyperbolic functions, etc. I would feel competent to design and write the user interface (e.g., keyboard input) sections of the calculator, but when it comes to even the floating-point arithmetic (let alone the scientific) functions, I'm totally lost. What do I do? Give up? Scour the different user groups software offerings, piecing together little routines from here and there? Take a course in applied numerical analysis and learn enough theory to do all the programming myself? No. Ideally, I'd browse through several software catalogs from commercial vendors, picking out the best math package for my needs. It would be relatively inexpensive, because of competition among the various vendors. And since the time I save by buying the package commercially is substantial, I'd consider it to be a larger-scale product (physically small though it may be). Finally, and most important, is the assurance of quality that I get – the programs that I buy should be backed up by a guarantee; if they don't work correctly, I can have the problem fixed by someone who's an expert at that sort of thing. In fact, any problems discovered by other customers of the supplier will be automatically corrected and reported to me.

I'm not pointing out anything surprising, though. What *is* surprising is that this lack of software is no one's fault. Why? The reasons will become clear as we examine the basis and effects of the vacuum more carefully.

The most obvious explanation for the software vacuum is the newness of our field. It's very easy to draw a parallel to the very early days of "real" computers, when every manufacturer was scrambling to produce hardware. The importance of software wasn't recognized, and it was simply left behind in the dust. Since then, however, software has become the major component, in cost, size, and complexity, of any large computer system, since nothing can be done without it.

The parallel is completed by noting that, now, the big "scramble" in microcomputers is to get out the hardware. People *do* sometimes learn from history, though. Microsystem suppliers have realized that naked machines are close to useless, and many are offering at least the bare bones system software mentioned earlier. The problem here is that many of us want to *use* our home or business system for something (e.g., as a powerful desk calculator) and don't necessarily *want to do all the* programming ourselves.

This situation is eased, for example, by the availability of a BASIC system from one major supplier. Unfortunately, for the "from scratch" hobbyist, the difficulty still remains, since software like BASIC is much more expensive as a *stand-alone* product (and is usually out of the price range of most of us). Again, since the average hobbyist would be lost when faced with the task of constructing his own BASIC system, he's thereby automatically cut out of the

You'll go nuts over our computers!



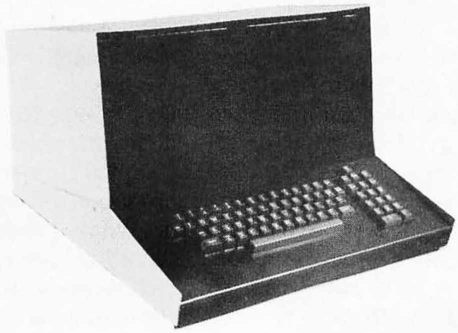
And the Sphere Computer System costs less than anyone else's terminal.

Completely intelligent micro-systems . . . that's what we offer. Just look at the features, and the prices. No compromising, with no short cuts! Recently at WESCON, SPHERE also demonstrated its new, full-color and B/W graphics terminal — and we have other new products, to be just as revolutionary as the SPHERE 1 SYSTEM was when we released it last June. SPHERE 'R & D' will keep ahead of your demands, no matter the state of the art. Take one look at our catalogue, then call or write us today.

KIT	ASM	
$860	$1400*	**SPHERE 1:** This system includes the Motorola 6800 microprocessor, 4K RAM, 1K EPROM (containing an EDITOR, ASSEMBLER, DEBUGGER, COMMAND LANGUAGE, CASSETTE LOADER, DUMPER, UTILITIES), a REAL-TIME CLOCK, plus 512 character video interface, with full ASCII keyboard and numeric/cursor keypad, power supply, chassis, manuals, and associated parts.
999	1499*	**SPHERE 2:** Includes all features of SPHERE 1, plus serial communications, and audio cassette or MODEM interface.
1765	2250*	**SPHERE 3:** Inlcudes all the features of SPHERE 2, plus memory totaling 20K which is sufficient to run full extended BASIC Language.
$6100	$7995*	**SPHERE 4:** Includes all of the features of SPHERE 3, except the cassette has been replaced by an IBM-compatable Dual Floppy Disk System. This system includes a Disk-operating system and BASIC Language and a 65 LPM line printer.
(various)		**OTHER SPHERE PRODUCTS:** Light pen option; full color and B/W video graphics system; low cost Dual Floppy Disk System; and full line of low cost peripherals.

*This ASSEMBLED SPHERE System includes the complete chassis, and video monitor as pictured below.

The Whole System:



SPHERE CORPORATION

791 South 500 West Dept. 1211
Bountiful, Utah 84010

(801) 292-8466

Suppose you could make *working* Xerox copies of CPU chips, how long would chip manufacturers be around?

Hobbyists can help generate commercial interest in producing software.

legacy of widely-available BASIC programs.

Now why can't we throw the blame on the CPU chip manufacturers? Because, clearly, their proper market is the OEMs (Original Equipment Manufacturers), and thus their main concern regarding software is to provide their OEM customers with developmental systems of hard and software, leaving the hobbyist nearly completely, if not intentionally, out of the picture.

What about the other potential suppliers of software? Look at the situation of a typical software house, or vendor. They must ask, "What kind of hobbyist market exists? How big is it, and how can it be reached? What kind of return on investment can be expected from a venture in this area?" These are tough questions for a company whose existence depends on successful marketing of software products, even though the answers might seem obvious to us. But keep in mind that the existence of BYTE is among the first indicators of a widespread hobbyist interest in computing, and we're only a few months old.

Furthermore, there's the problem of proprietary programs, a problem hard to appreciate from the hobbyist's point of view. It stems from the softness of software: Hardware can only be physically in one place at one time, but software, because its copying cost is (relatively) so low, can virtually be in many places at once. For example, an institutional computer installation signs a proprietary contract when it buys a program product. Such contracts typically restrict use of the software to one computer system or customer site. This is one of the major reasons the institution is not likely to lend or sell such a product to other installations. But a hobbyist, even though he might sign a similar contract, is much more likely to help out a friend by swapping a program for others, or by lending a copy to someone else. And this is deadly poison for any sort of commercial market. To take a concrete, if absurd, illustration, suppose you could make *working* Xerox copies of CPU chips – how long would the chip manufacturers be around?

Another simple fact of proprietary-program life is that such programs cost a lot to develop and market. Unless the market is large, this means high prices, which a big computer installation can usually justify, but which an individual just can't afford. It's true that there's a difference in scale between large and micro system software, but both, in their own spheres, are costly to commercially develop and maintain. So, the software vendors may simply not have been able, even if willing, to enter the microcomputer software market.

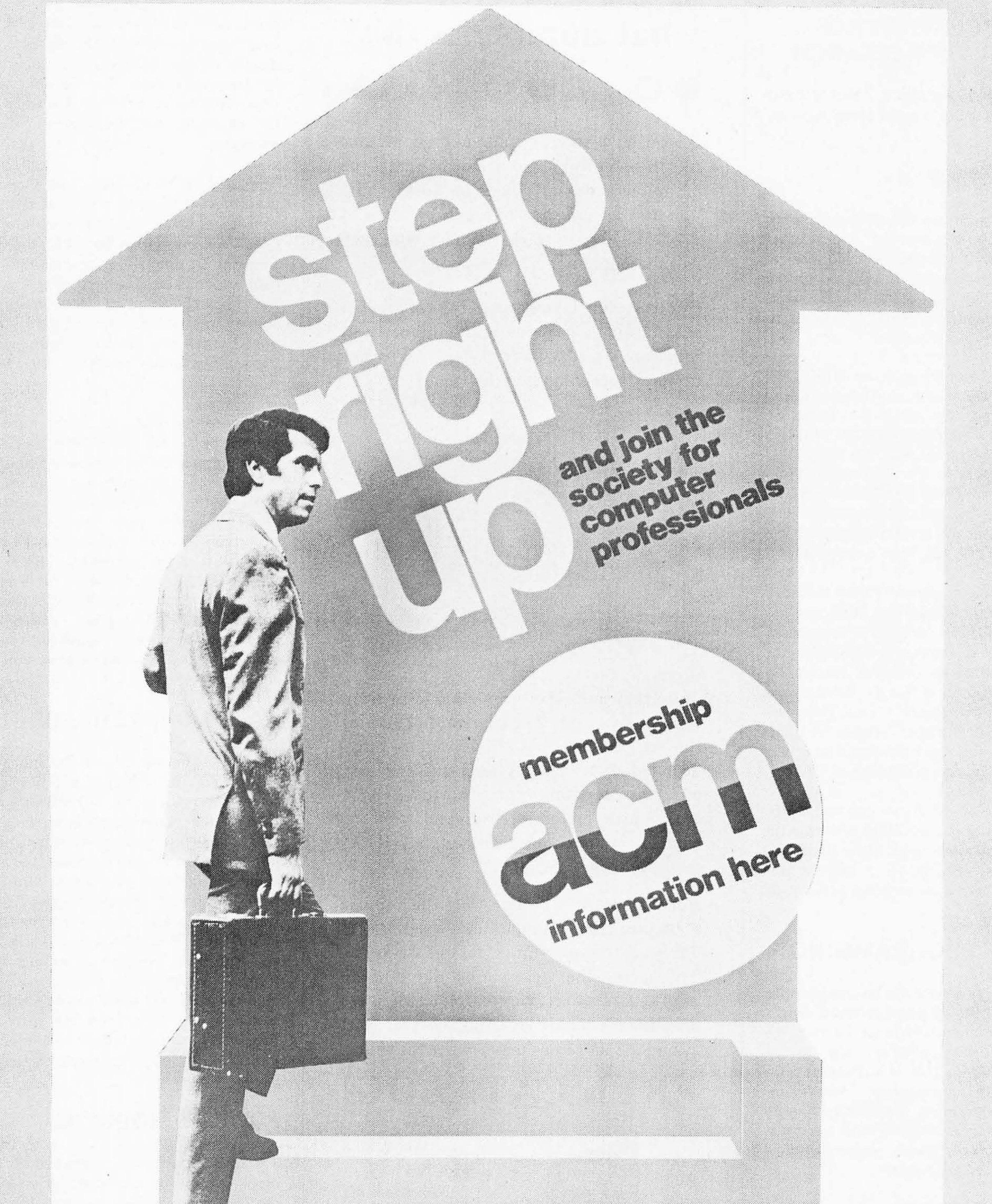
Continuing with our search for potential software suppliers, we arrive at the grass roots level: The home experimenter who's developed a good product with long hours and minimal tools. Why can't he go it alone? Well, he's usually operating on a small budget and thus can't mount any sort of larger marketing, packaging, or shipping effort. True, you might say, but couldn't he sell his product locally, at a neighborhood level? Yes, he could, but then he's not reaching the majority of people who are interested in his product, namely, *us*. Besides, how can he commercially support any product on even a medium scale, when such support might involve, aside from all the developmental and promotional work and expense, handling dozens of trouble reports in a single week? He'd no longer be a hobbyist, but a full time one-man software house, and that puts him out of the grass roots class.

So, with our search ended, and no culprit in sight, what can be done about the software vacuum?

First of all, those of us in the personal computing field who have professional contacts can urge existing and potential software vendors to look hard at the hobbyist markets. *When the number* of potential users of a package is multiplied by the profit margins to be expected, such software vendors should be economically viable. Whenever there is a demand for a product, a free market will tend to fill that demand. It will be interesting to see what the market produces in system software for small computers.

Second, hobbyists can help generate commercial interest in this vital area in several ways. One is to make the software vacuum a topic for discussion at local computer club meetings. Another is to organize software trading posts in the newsletters which are very much a part of the hobby. Still another is to write manufacturers urging advanced software products to match the extremely high level of today's hardware technology. If you feel eloquent, write a letter to the editor of BYTE on the subject.

The first step in filling a need is identifying that the need exists. Hopefully these thoughts will start some BYTE readers off in a direction which will lead to commercially marketed mass produced software which can be plugged into one of the several microprocessor architectures which are on the market. ■


step
right
up
and join the
society for
computer
professionals
membership
acm
information here

The MODULAR MICROS from MARTIN RESEARCH

Here's why the new *MIKE 2* and *MIKE 3* are the best values in microcomputers today!

8008 OR 8080

Martin Research has solved the problem bothering many potential micro users .... whether to go with the economical 8008 microprocessor, or step up to the powerful 8080. Our carefully designed bus structure allows either processor to be used in the same system!

The *MIKE 3* comes with an 8080 CPU board, complete with crystal-controlled system timing. The *MIKE 2* is based on the 8008. To upgrade from an 8008 to an 8080, the user unplugs the 8008 CPU board and plugs in the 8080 CPU. Then he unplugs the 8008 MONITOR PROM, and plugs in the 8080 MONITOR PROM, so that the system recognizes the 8080 instruction set. That's about it!

If the user has invested in slow memory chips, compatible with the 8008 but too slow for the 8080 running at full speed, he will have to make the 8080 wait for memory access—an optional feature on our boards. Better still, a 4K RAM board can be purchased from Martin Research with fast RAM chips, capable of 8080 speeds, at a cost no more that you might expect to pay for much slower devices.

In short, the *MIKE 2* user can feel confident in developing his 8008 system with expanded memory and other features, knowing that his *MIKE 2* can be upgraded to a *MIKE 3*—an 8080 system—in the future.

EASE OF PROGRAMMING

Instructions and data are entered simply by punching the 20-pad keyboard. Information, in convenient octal format, appears automatically on the seven-segment display. This is a pleasant contrast to the cumbersome microcomputers which require the user to handle all information bit-by-bit, with a confusing array of twenty-odd toggle switches and over thirty red lights!

A powerful MONITOR program is included with each microcomputer, stored permanently in PROM memory. The MONITOR continuously scans the keyboard, programming the computer as keys are depressed.

Say the user wishes to enter the number *135* (octal for an 8008 OUTPUT 16 instruction). He types *1,* and the right-hand three digits read *001.* Then he presses *3,* and the digits say *013.* Finally he punches the *5,* and the display reads *135.* Notice how the MONITOR program

(Continued in column 3.)

shifts each digit left automatically as a new digit is entered! The value on the display is also entered into an internal CPU register, ready for the next operation. Simply by pressing the *write* key, for example, the user loads *135* into memory.

The MONITOR program also allows the user to step through memory, one location at a time (starting anywhere), to check his programming. Plus, the Swap Register Option allows use of the interrupt capabilities of the microprocessor: the MONITOR saves internal register status upon receipt of an interrupt request; when the interrupt routine ends, the main program continues right where it left off.

We invite the reader to compare the programmability of the *MIKE* family of microcomputers to others on the market. Notice that some are sold, as basic units, *without any memory capacity at all.* This means they simply cannot be programmed, until you purchase a memory board as an "accessory." Even then, adding RAM falls far short of a convenient, permanent MONITOR program stored in PROM. Instead, you have to enter your frequently-used subroutines by hand, each and every time you turn the power on.

EASY I/O INTERFACE

The *MIKE* family bus structure has been designed to permit easy addition of input and output ports. A hardware interface to the system generally needs only two chips—one strobe decoder, and one latching device (for output ports) or three-state driving device (for inputs). A new I/O board can be plugged in anywhere on the bus; in fact, all the boards in the micro could be swapped around in any position, without affecting operation. I/O addresses are easy to modify by reconnecting the leads to the strobe decoder (full instructions are provided); this is in marked contrast to the clumsy input multiplexer approach sometimes used.

POWER & HOUSING

The micros described to the left are complete except for a cabinet of your own design, and a power supply. The basic micros require +5 V, 1.4 A, and −9 V, 100 MA. The 4K RAM board requires 5 V, 1 A. A supply providing these voltages, and ±12 V also, will be ready soon.

OPTIONS

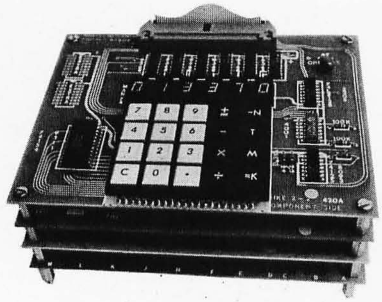
A number of useful micro accessories are scheduled for announcement. In addition, the *MIKE 3* and *MIKE 2* may be purchased in configurations ranging from unpopulated cards to complete systems. For details, phone, write, or check the reader service card.

QUICK....
what number is this?

● ○   ○ ● ○   ● ○ ●

If you have to read your microcomputer like this--bit by bit, from rows of lights--the computer's making *you* do *its* work. And if you have to use rows of toggle switches to program it, you might wonder why they call the computer a labor-saving device!

Contrast the layout of a typical pocket calculator. A key for each number and function; six easy-to-read digits. Why not design microcomputers like that?



Here they are! The *modular micros* from Martin Research. The keyboard programs the computer, and the bright, fully-decoded digits display data and memory addresses. A Monitor program in a PROM makes program entry easy. And, even the smallest system comes with enough RAM memory to get started!

Both the *MIKE 2* system, with the popular 8008 processor, and the 8080-based *MIKE 3* rely on the *same* universal bus structure. This means that accessories--like our 450 ns 4K RAM--are compatible with these and other 8-bit CPUs. And, systems start at under $300! For details, write for your...

FREE CATALOG!



MIKE 2 MANUAL... This looseleaf book includes full information on the *MIKE 2* system, with schematics.

Price for orders received by November 15, 1975... **$19**
Includes a certificate worth $10 towards a *modular micro* system, good 90 days. *(Offer valid, USA only.)* After 11/15: $25.

modular micros | **martin research**

Martin Research / 3336 Commercial Ave.
Northbrook, IL 60062 / (312) 498-5060

BYTE'S BITS

The Secret of Unraveling Wire Wrap Boards

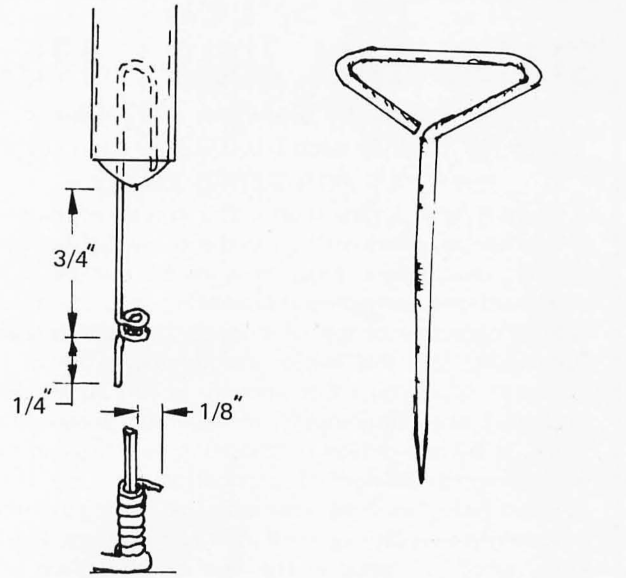


Fig. 1. A homebuilt tool to loosen and remove wire wrapped connections.

A lot of home brew computer people are obtaining surplus printed circuit cards and back planes that have been wire wrapped. In order to use the components from such boards the wires must be carefully pulled off the wrapping posts. Without a special tool it is difficult to remove wire while preserving the integrity of the delicate wrapping posts. Since many BYTE readers do not have the commercial dewrapping tools, removal of wire is for them a nasty and time consuming job at best.

I have devised a simple tool which makes this task much easier, made from items normally found around a well equipped workshop. The tool, shown in Fig. 1, is constructed from a paper clip and a drapery hook. Simply follow the instructions.

1. Obtain a standard paper clip, an Exacto knife holder, and a thin sharp pick of sorts, such as a drapery hook.
2. Bend the paper clip and drapery hook as shown.
3. Place the paper clip in the Exacto blade holder. Now you are ready to dewrap.
4. Using the pick, flick the wire from the post at least an eighth of an inch.
5. Place the paper clip bit onto the wire wrap post and slip it down until the end of the paper clip is just past the wire.
6. Rotate the blade holder in the direction which unravels the wire until the entire wire is loose.

by
Richard J. Lerseth
8245 Mediterranean Way
Sacramento CA 95826

WORD HUNT

S	L	I	N	K	E	T	Y	B	E	T	E	C	A	F	R	E	T	N	I	M
S	B	A	U	D	C	F	O	R	T	R	A	N	Y	R	O	M	E	M	N	S
E	R	I	W	E	I	O	R	A	I	A	E	R	P	R	I	O	R	I	T	Y
R	S	O	U	R	C	E	L	N	X	P	D	L	I	T	E	R	A	L	E	N
D	R	L	E	D	E	T	U	C	E	X	E	I	B	U	F	F	E	R	R	T
D	E	V	I	C	E	N	Y	H	T	R	A	N	S	M	I	T	T	E	R	A
A	D	E	S	T	I	N	A	T	I	O	N	C	H	K	E	T	R	A	U	X
L	A	C	O	F	L	A	G	E	M	E	G	A	R	O	T	S	I	D	P	S
R	E	T	N	U	O	C	T	A	L	G	L	O	B	A	L	I	S	C	T	U
O	G	O	A	D	D	R	R	A	M	O	T	A	B	L	E	L	B	A	N	B
T	A	R	C	P	E	G	M	I	C	R	O	P	R	O	C	E	S	S	O	R
A	T	B	M	U	O	U	E	A	I	S	L	B	P	F	D	T	L	Y	I	O
L	E	U	E	R	S	U	T	A	T	S	P	A	I	A	S	C	A	N	T	U
U	J	U	P	K	S	I	S	Z	S	R	B	S	H	L	T	I	R	C	C	T
M	Q	R	C	N	O	F	Y	H	O	E	I	I	C	S	A	N	E	H	U	I
U	T	E	T	N	I	M	S	M	N	C	N	C	T	E	C	O	H	R	R	N
C	H	T	T	L	D	E	B	U	G	O	A	L	E	A	K	M	P	O	T	E
C	D	N	E	N	A	A	U	L	A	R	R	O	F	S	L	E	I	N	S	D
A	A	I	A	P	T	H	S	C	I	D	Y	B	L	O	G	N	R	O	N	O
P	O	O	L	C	A	S	O	E	D	O	C	O	R	C	I	M	E	U	I	I
A	L	P	H	A	N	U	M	E	R	I	C	C	V	Y	A	L	P	S	I	D

Hidden in the matrix are 96 words and abbreviations associated with computers in one way or another. Find a word, circle it or color it in, and cross it off the list. Words may be forward, backwards, up, down, or diagonal, and are always in a straight line, never skipping letters. However, some letters are used more than once. After circling all the words on the list, the unused letters (14 total) will spell out a secret message. Beware: Some buzzwords occur in the matrix, but are not on the list!

Good luck and good hunting . . . it may take a while!

ACCUMULATOR
ADD (not in address)
ADDRESS
ALPHANUMERIC
AND
APL
ASCII
ASSEMBLER
ASYNCHRONOUS
BASIC
BATCH
BAUD
BCD
BET
BINARY
BOOLEAN
BRANCH
BUFFER
BYTE
CHECKSUM
CHIP
CLOCK
COBOL
COUNTER
DATA
DEBUG
DESTINATION
DEVICE
DIAGNOSTIC
DIODE
DISK
DISPLAY
DTL
ECL
EXECUTE
EXIT
FALSE
FETCH
FILE
FLAG
FOCAL
FORMAT
FORTRAN
GATE
GLOBAL
HALT
INPUT
INSTRUCTION
INTERFACE
INTERRUPT
JUMP
LED
LINK
LIST
LITERAL
LOAD
LOCATION
LOG
LOOP
LSI
MATRICES
MEMORY
MHZ
MICROCODE
MICROPROCESSOR
MNEMONIC
MOS
OCTAL
PCB
PERIPHERAL
PLOT
POINTER
PRIORITY
PROGRAM
PROM
QUEUE
RAM (not in program)
READ
RECORD
ROL
SOURCE
STACK
STATUS
STORAGE
SUBROUTINE
SUBSYSTEM
SYNTAX
TABLE
TRANSMITTER
TRAP
TTL
VCC
VECTOR
VERIFY
WIRE
WORD

Logic Probes - Hardware Bug Chasers

by
Alex. F. Burr
Physics Dept. Box 3D
New Mexico State University
Las Cruces NM 88003

While an oscilloscope or voltmeter can be used with digital circuits, a logic probe is much less expensive if built from an appropriate kit.

Digital logic, whether used in an 8080 microprocessor or as the TTL chips that can be used to make a processor, is, at least in theory, clean and simple because only two states are possible. Any point in even the most complicated circuit is either HIGH or LOW. However this very simplicity encourages the design of large and complicated circuits. While the chance of anything going wrong at any one point is small, the accumulated chances of many points means that sooner or later the experimenter is going to have to hunt for sources of trouble.

In the case of analog circuits, when trouble develops, you get out the oscilloscope or voltmeter and start looking for places which have waveforms or voltages not meeting the specifications. These instruments can be used to troubleshoot digital circuits too. The oscilloscope is particularly useful if you have timing problems, but usually they give you too much information and may just confuse the issue. The voltmeter may tell you that the voltage on pin 8 is 3.9 but, because most IC failures show up as a node stuck either HIGH or LOW, really all you need to know is that on pin 8 there is a HIGH. That single bit (literally) of information can be obtained with an instrument a lot smaller and less complicated than a voltmeter.

That instrument is the *logic probe.* In its simplest form it is just a state indicator with a sharp point. When the point is placed on any pin of an IC, the probe will indicate whether a LOW or a HIGH is present at that point. And with digital logic that is usually all the information you need.

Logic probes can detect a surprising number of different defective conditions. Fig. 1 illustrates some of the uses to which a probe can be put. Of course, just as voltmeters come with a variety of capabilities and prices, so do logic probes.

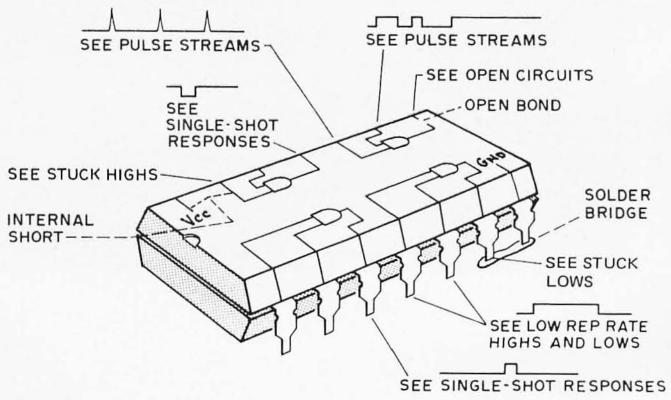
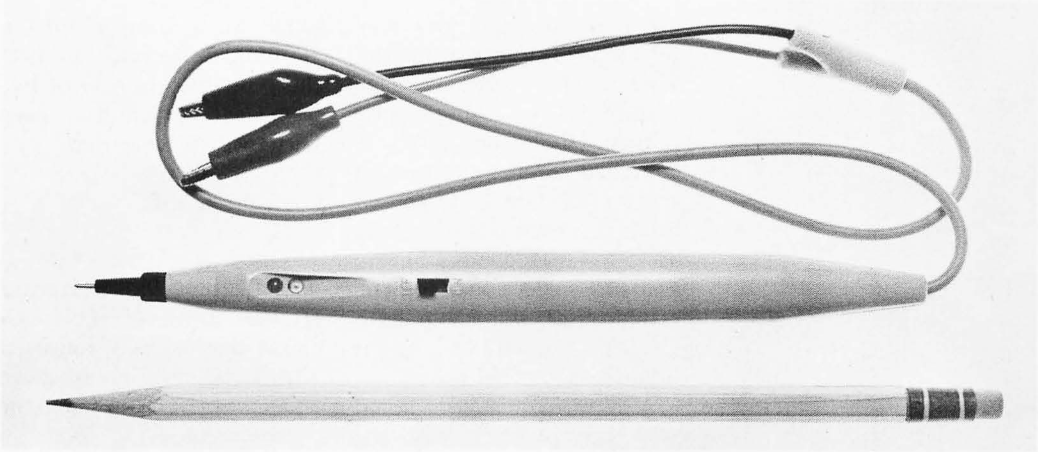


Fig. 1. Some of the uses of logic probes and the malfunctions which they can detect.

Commercial Logic Probes



The E & L Instruments logic probe is compact, with the two indicator LEDs visible toward the left in this photo.

One of the first developed was the Hewlett Packard 10525T logic probe. It is a marvel of compactness and versatility, all carefully human-engineered. Basically it consists of a white light which goes out when the probe is placed on a LOW and comes on when the probe is placed on a HIGH.

Simple -- yes indeed; but it does much more. What if the point tested is open circuited, or the level is just plain bad, neither HIGH or LOW? Then the light glows at half intensity. What if a pulse comes along that is too short to excite the indicator light? Then a pulse stretcher takes over. Pulses with a width of between 10 ms and 0.05 seconds are stretched to 0.05 seconds in length. What if the pulses come so fast that the eye cannot distinguish one from the next? All pulse streams with a repetition rate between 10 Hz and 50 MHz cause the lamp to blink at a 10 Hz rate. All this capability is enclosed in a probe about six inches long and one-half inch in diameter. The light is placed near the tip in such a way that it can be seen no matter how the probe is rotated. Thus you can easily see both the point of the probe and the indicator at the same time. Power is supplied to the probe by a well protected single cord which is attached to a source of 5 V dc at 60 mA.

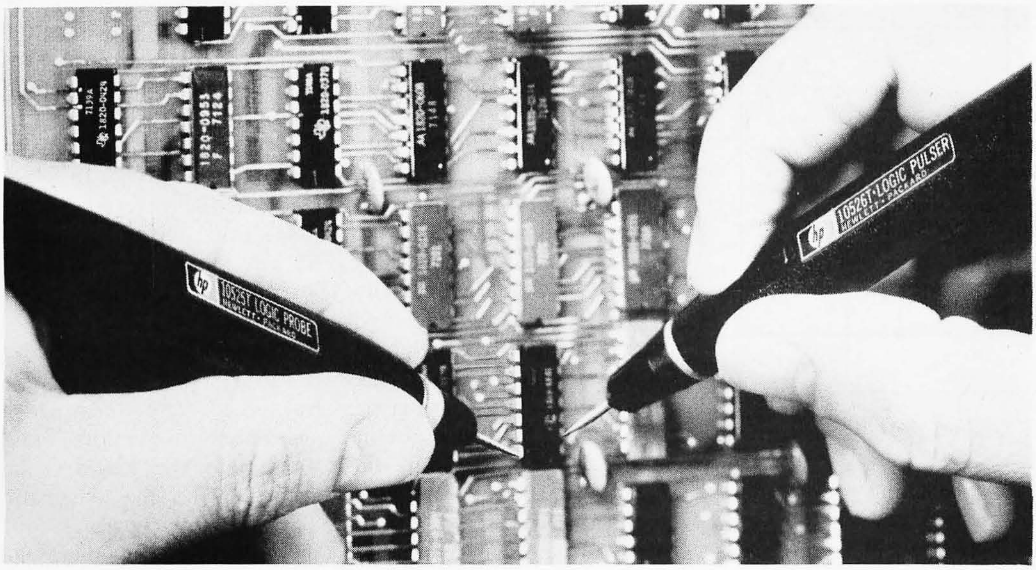
The input impedance is greater than 25k Ohms in both the HIGH and LOW state (less than one low power TTL load). The input is well protected against operator error. The probe will stand ±70 volts continuously and ±200 volts intermittently as well as 120 V ac for 30 seconds. The power input is internally protected from +7 to −15 V dc as well as power lead reversal. The only catch is the price, which even with a recent reduction is $65.

There are, however, other less expensive probes. Two of these are distributed by E and L Laboratories. Their model 340 is a logic probe and pulser combined into one instrument. The model 320 is a logic probe only, designed to give maximum information about the state of the node being tested. Both probes are well constructed, a little over 6 1/2 inches in length and half an inch in diameter. Both come with two different probe tips and handy carrying cases.

The model 340 has two LED indicator lights. In operation the two leads from the probe are connected to the 5 V dc supply and the probe tip applied to the IC lead to be tested. If that node is HIGH, the red LED lights up brightly. If that node is LOW, neither LED is lit. The green LED is used to detect pulses which do not last long enough to give a useful indication on the red LED. It lights for 0.1 seconds (just long enough to see) whenever a single pulse wider than 50 nanoseconds is applied to the probe tip. In the probe tested for this article, when a voltage increasing from zero was applied to the tip, the red LED lit when the voltage was greater than 1.5 V; just what the specifications called for.

The model 320 is a little more versatile as an indicating instrument, but it lacks the ability to generate pulses that the model 340 has. It too has two LED indicators, one red and one green. In operation, the two power leads are connected to the 5 V dc power supply and the tip to the node under test. The specifications say that if the voltage at the node is less than 0.7 volts the green LED will be lit. If the voltage is greater than 2.4 volts, the red LED will be lit. In practice the specifications are closely followed. The LEDs may glow dimly at voltages just a few tenths higher or lower than the specified voltages. But the dividing line between lit and not lit states is remarkably sharp.

A special feature of this probe is the pulse storage



The Hewlett Packard 10525T logic probe and 10526T pulser.

capability brought into play by a small switch near the tip. When the pulse storage feature is on, a short pulse (either HIGH or LOW) is stretched so that it turns on the appropriate LED to full brightness even if it is as short as 50 nanoseconds. Square and sine waves appearing at a tested node will cause both LEDs to have equal brightness.

The main difficulty noted with this probe is with the green LED. It is somewhat dimmer than the red LED and the lens diffuses the spot of light generated less well so that in bright room light it is sometimes hard to tell whether or not the green LED is lit. This fact would make the determination of the duty cycle of a chain of pulses by a brightness comparison between the LEDs much more difficult than the instruction booklet suggests.

"Nodes" are places in a circuit – such as the pin of an IC – where you might want to test the logic level using the probe.

Even the E and L Laboratories probes are expensive ($35 and $25); although they are more convenient than, and certainly in the same price range as, a good voltmeter. Nothing, however, can beat the cost effectiveness of two probe kits which have been fairly widely advertised.

Logic Probe Kits

One of these kits is manufactured by Chesapeake Digital Devices. This kit allows one to easily construct a probe which uses red, green and yellow LEDs to signal the presence of logic levels in digital circuits.

The kit goes together in a very short time with the aid of very complete assembly instructions. The whole probe fits into a well constructed case, a little over six inches long and slightly less than one inch in diameter. There are only three resistors, three LEDs, one transistor, and a 74S00 integrated circuit to solder onto the clearly marked printed circuit board.

In operation the green LED is brightly lit on a LOW, the red LED is brightly lit on a logic HIGH, while the yellow LED lights on an open circuit or a level between a true HIGH or LOW. A slow pulsing condition will be indicated by alternate flashing of the red and green LEDs. A fast pulsing condition will be indicated by the simultaneous activation of the red and green LEDs. The dividing line between these last two conditions is about 20 Hz, depending on the eye of the user.

The biggest difficulty with the kit was the circuit board. The copper leads had not been tinned and were oxidized, making them a bit difficult to solder; especially if the builder was concerned that he not use so much heat for so long as to damage the components. The clear plastic tube into which the circuit board with its LEDs slide did crack on assembly and the green LED was open but these difficulties were easily remedied and the result was a handy logic probe at a price significantly less than any assembled probe.

A Unique Probe

A particularly inexpensive kit is the one sold by James Electronics for $9.95 including postage and case. It is unique in that it uses a MAN 3 seven segment readout which gives a 1 for a HIGH a 0 for a LOW and a P for a pulse train – all this in a compact package measuring five inches long and one inch in diameter.

The circuit diagram for this intriguing probe is given in Fig. 2. The 2N2222 input transistor drives the chip, IC1, which in turn causes the appropriate segments of the MAN 3 to light. The chip was custom made for James Electronics by National Semiconductor and contains a proprietary circuit which was laid down by a $500 master mask.

The kit comes in a very impressive package which was carefully designed to protect the contents from rough handling by the U.S. Postal Service. The parts, which include the case and a custom glass epoxy printed circuit board, are of high quality and are not your usual cheap imports. Because most of the parts are in the 14-pin chip which is the heart of the probe, the kit goes together quickly and easily for the experienced builder (about one hour to solder all the parts to the board). There are no explicit devices for overload or reverse voltage protection. The probe draws 65 mA from any convenient 5 V point on the circuit under test.

The inexperienced builder is going to have trouble because the complete assembly instructions say, "Assemble the Logic Probe according to the schematic diagram and board layout shown below." The end. One has to have pretty sharp eyes

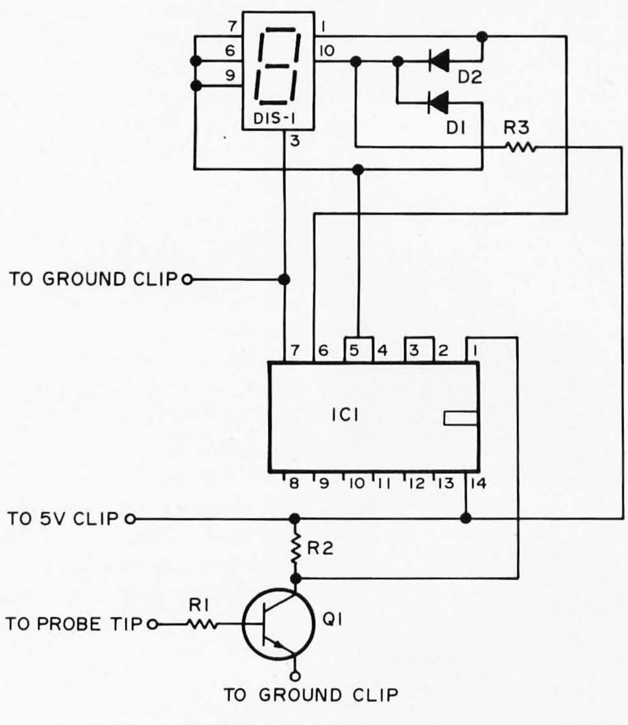
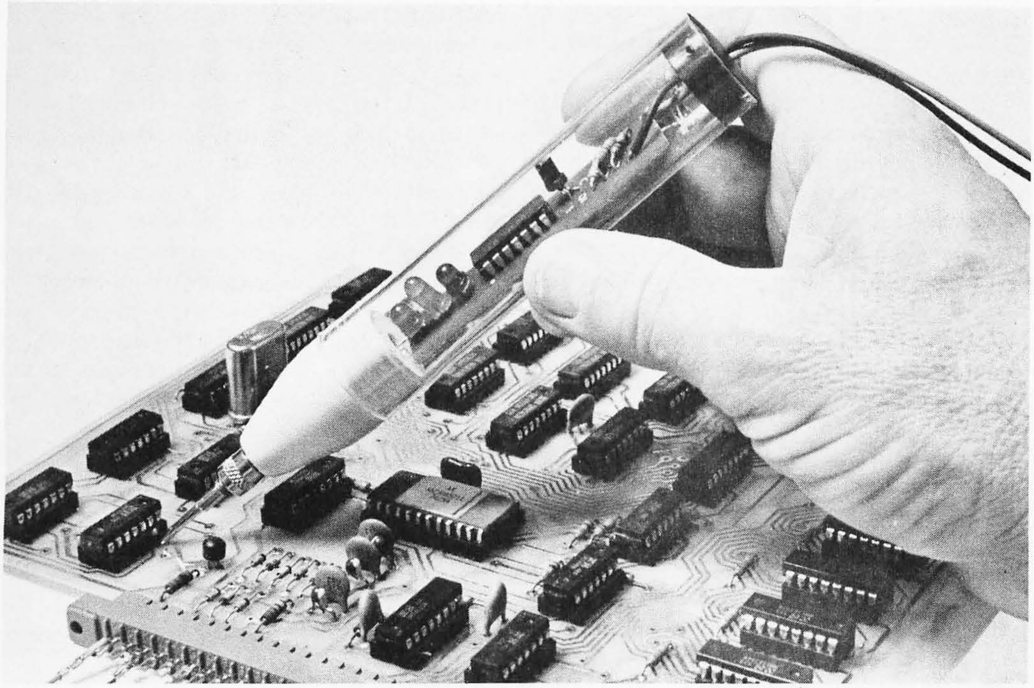


Fig. 2. Circuit diagram for the James logic probe kit.



A kit logic probe shown in action testing a printed circuit board.

A logic clip is like 16 binary voltmeters in a neat little package.

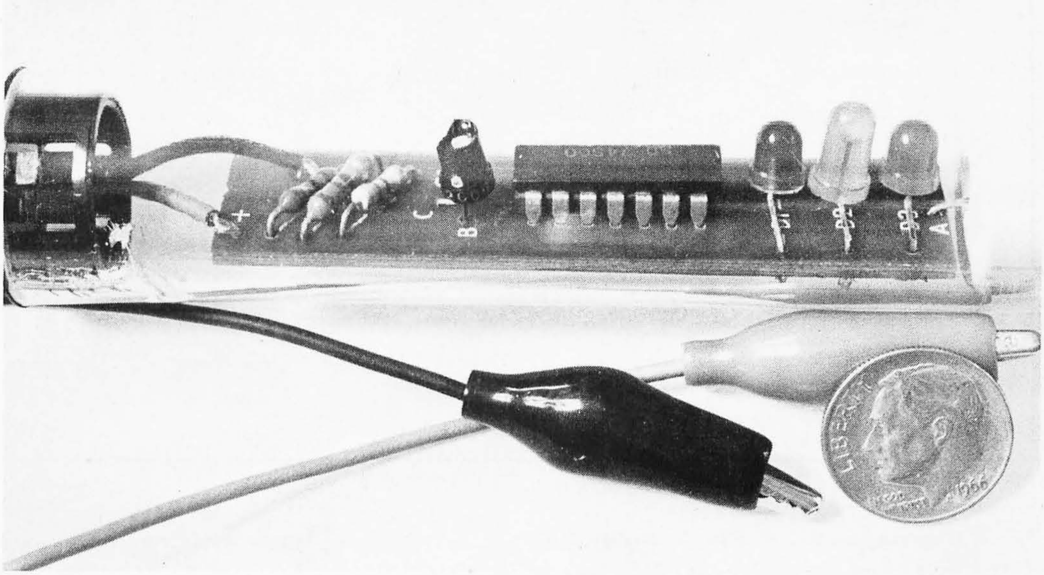
to orientate the IC, transistor and readout correctly. Even then you might miss the two jumpers that go on the circuit board. The circuit board also could have been laid out more efficiently so that the drastic bending of the MAN 3 leads would have been avoided.

There is one serious defect. It is more serious from the theoretical than the practical point of view. That defect concerns the input level at which the indicator switches from 0 to 1. That level is 0.65 volts; but the specifications for TTL logic say that the maximum voltage that the logic is guaranteed to interpret as LOW is 0.8 volts. Thus the probe would indicate a HIGH on a node which the logic would interpret as a LOW. This defect is of lesser practical importance because it is the unusual LOW which will have a voltage greater than 0.6 V. Indeed the usual gate input is only a very few tenths of a volt above ground at the most when it is LOW. Nevertheless it is a bit disconcerting to have the probe give a wrong reading even if it does so only under unusual circumstances. After all, it is under unusual circumstances that the probe is most often used.

Logic Clips?

The probes which have been discussed so far all investigate one pin of the IC at a time. There are some instruments which will do much more. These are called logic clips and are not really probes but will give you the same type of information. They are extremely handy service and design tools. They clip onto TTL DIP ICs and instantly display the logic states of all 14 or 16 pins. Each of the clip's 16 LEDs independently follows level changes at its associated pin: A lighted diode corresponds to a HIGH.

The logic clip's real value is in its ease of use. It has no controls to set, needs no power connections, and requires practically no explanation as to how it is used. The clip has its own gating logic for locating the ground and +5 volts Vcc pins and the buffered inputs reduce circuit loading. Simply attaching the clip to a TTL dual inline package makes all



A detail of the Chesapeake Digital Devices logic probe board. The three LEDs are at the right in this picture, with the 74S00 IC in the center.

the logic states visible at a glance. The clip is, in effect, 16 binary voltmeters. When used with some means of pulsing a complicated circuit slowly, sequential logic states like shift registers come alive – each state change is immediately visible.

The most popular clips are made by Hewlett Packard and Circuit Specialties. Unfortunately they have one big drawback – price. They cost from $75 to $85 each and will not be discussed further here.

Summary

Table 1 summarizes all the information that has been given here and presents some new facts about each of the logic probes discussed. By scanning this table you ought to be able to determine which probes have the features you need and the ones you can afford. The following comments are based on personal experience with each of these probes, but that experience has been rather limited.

The Hewlett Packard probe is obviously the best. It should be; it certainly costs significantly more. It will work under a wide range of conditions and it is carefully made. For the extra money you get wide frequency range, tight specifications, and vastly superior handling of pulse trains. The construction is first class and includes such extras as a compact BNC plug on the power cable (which, of course, is not so good if your breadboarding system does not have a BNC jack to supply that power).

The E and L probes (340 and 320) are imported from Japan. They are very well constructed and have the little extras like plastic carrying cases and different probe tips that the better Japanese manufacturers like to include with their products. The 320 is a better logic probe than the 340. It is less expensive and it handles pulse trains and logic levels in a better and more revealing way. Of course, it does not have the pulse generating capabilities of the 340.

The professional logic designer will want to get one of these three probes. They may be a bit expensive for the serious hobbyist. In that case one of the two kits would be satisfactory.

Both kits went together easily and rapidly. The CDD kit is much more revealing about the state of the logic under test and has superior assembly instructions. The James kit has better quality parts and is cheaper.

In any case the serious worker in digital logic and computers, whether a professional or a serious hobbyist, will find one of these probes a valued addition to his collection of test equipment. ■

Table 1. Characteristics of logic probes.

Probe	HP 10525T[1]	340[2]	320[2]	CDD[3]	James[4]
Operating Voltage	5 ± 10% V	5 ± 10% V	5 ± 10% V	5 ± 10% V	5 V
Current	60 mA	100 mA	80 mA	40 mA	65 mA
Frequency Response	50 MHz[5]	12 MHz	12 MHz		
Input Impedance	>25 kΩ	50 kΩ	100-600 kΩ		
Min. pulse width	10 ms[6]	50 ms	50 ms		See[11]
Levels OPEN	half intensity	no lights[7]	no lights[9]	yellow[10]	
HIGH	on >2±0.2 V	red >1.5 V	red >2.4 V	red >2.5 V	1 >0.7 V
LOW	off $<0.8^{+0.2}_{-0.4}$ V	no lights	green <0.7 V	green <1 V	0 <0.7 V
Size	6" x 0.5" dia.	6.6" x 0.6" dia.	6.6" x 0.6" dia.	6" x 1" dia.	5 x 1" dia.
Overvoltage protection	excellent	reasonable	reasonable	none	none
Price	$65	$35[8]	$25	$15[12]	$10[12]

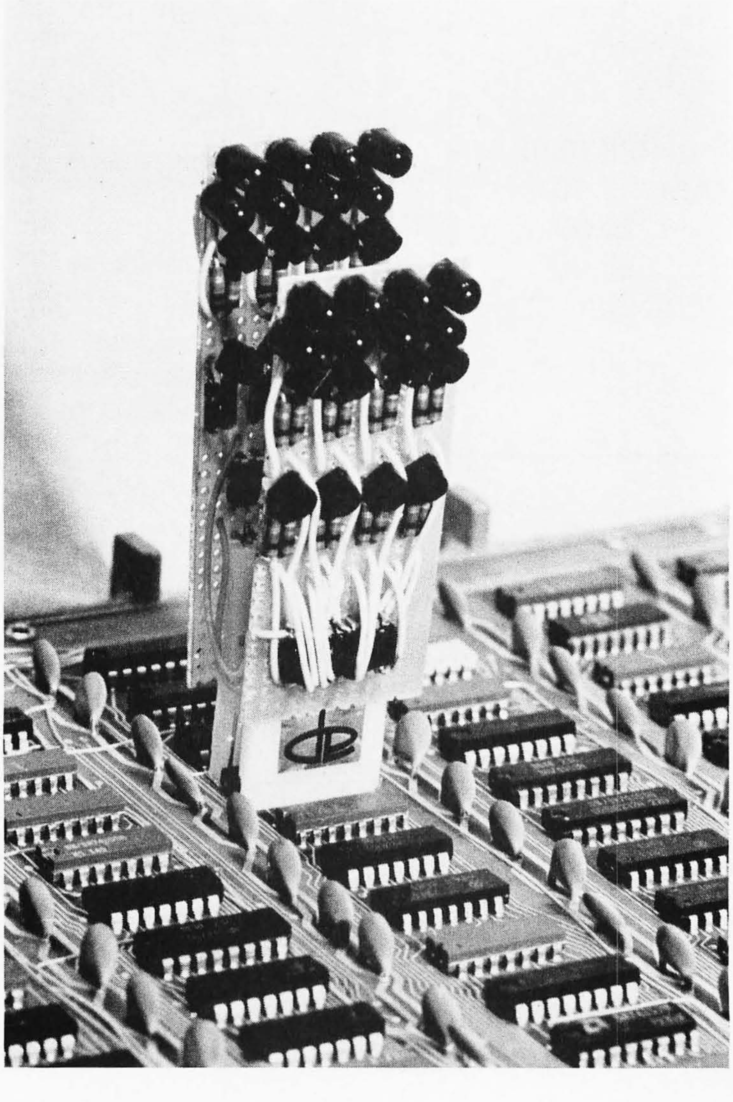
Notes:

[1] Hewlett Packard, Palo Alto CA 94304.
[2] E and L Instruments Inc., 61 First St., Derby CT 06418.
[3] Chesapeake Digital Devices, Inc., Box 341, Havre de Grace MD 21078.
[4] James Electronics, Box 822, Belmont CA 94002.
[5] Pulse trains faster than 10 Hz cause the lamp to flash at a 10 Hz rate.
[6] Pulses between 10 ms and 0.05 seconds are stretched to 0.05 seconds.
[7] Short pulses indicated by green LED.
[8] Single pulse generator contained in probe.
[9] Switchable pulse stretcher for short pulses.
[10] Yellow LED is also lit if voltage is between HIGH and LOW.
[11] Indicator reads P for pulse trains > 20 Hz.
[12] Kit price.

Powerless IC Test Clip

circuit by
John Errico
5 Richard Rd.
Hudson MA 01749

written by
Robert Baker
34 White Pine Dr.
Littleton MA 01460



This test clip operates like the expensive, commercially available clips selling for $85 or more without requiring batteries or external power. All types of ICs may be tested (TTL, DTL, MOS, etc.) and LEDs are used to indicate the logic state of each pin being tested.

The heart of the test clip is a Texas Instruments TID125 diode array which costs about $3.75. Two diode arrays are used to determine the pin with the highest voltage (Vcc) and the pin with the lowest voltage (ground). These pins are then used to power the LEDs on the test clip itself, thus taking power from the IC on the board and eliminating the need for an external or separate supply. The circuit is straight forward and may be expanded to make a 24- or 40-pin test clip. The larger test clip, however, may be difficult to use due to the size of the LED display.

The basic IC clip is a standard item available from AP Products Inc., Box 110-Z, Painesville OH 44077. The 16-pin clip is part number 923700 (TC-16) and sells for $5.75 each.

The diode arrays are 14-pin dip packages and were chosen to make the test clip more compact. To cut down the cost, 16 general purpose silicon diodes may be used in place of each diode array IC. The transistors used to drive the LEDs may be any NPN transistor capable of handling the LED current. Any small size LED may be used; however, the 1k resistance value may have to be changed. Choose a value which gives about 2 mA current through the LED; this should give sufficient brightness without loading down the circuit supply.

Construction is very simple and parts layout is not critical. Use a small piece of 0.1" grid perforated board bolted to each side of the IC clip to mount components on. Try to keep the overall physical size of the boards as small as possible to make the finished test clip easier to handle. The LEDs should be mounted along the top edge of the perforated boards so they are visible from above the clip when it is attached to an IC. I would suggest wrapping a small piece of dark tape or using a short piece of dark tubing around each LED to improve visibility of the finished LED display. One of the TID125 diode arrays is mounted on each piece of perforated board along with the associated resistors and transistors, positioned wherever convenient. Remember to run two wires between the two perforated boards to connect the Vcc and ground outputs of the diode arrays together. These wires should be stranded to withstand the movement of opening and closing the test clip when in use.

Using the test clip is the simplest part of all. Just clip it over the desired IC. Don't worry about how to position the test clip on the IC; pin 1 may be at either end and the test clip will still work properly. With the test clip installed on an IC package the LEDs will indicate the logic level of each pin:

ON = Logic 1 (HIGH) or Vcc pin
OFF = Logic 0 (LOW) or ground pin

On 14-pin ICs disregard the two pins not attached.

Who said building an IC test probe is hard? ■

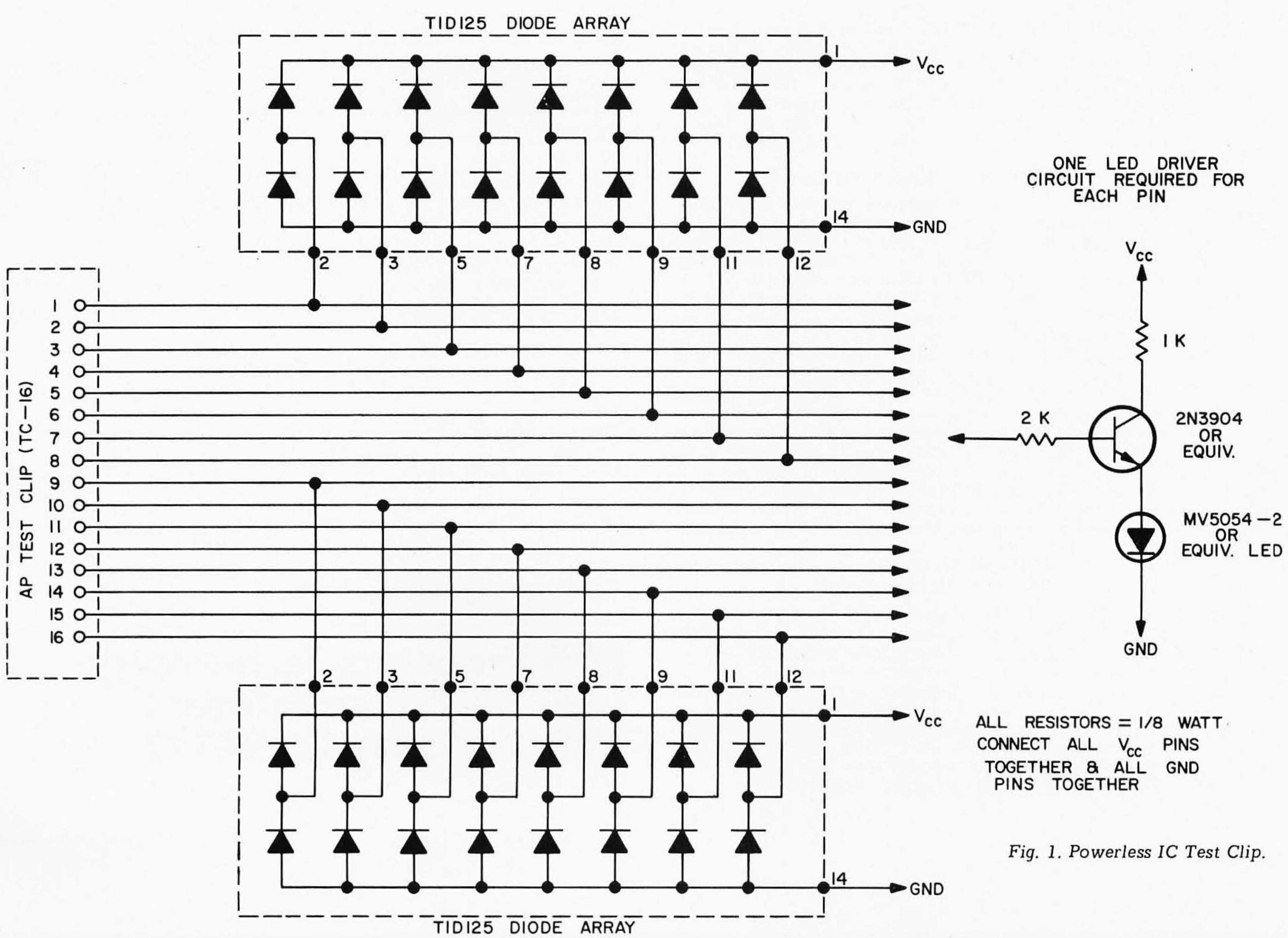


Fig. 1. Powerless IC Test Clip.

8800 SOFTWARE!

WE HAVE ALTAIR COMPATIBLE 8080 SOFTWARE AND FIRMWARE MODULES!

If you haven't used our Assembly Language Operating System you have been missing a wonderful experience. We have found the ALOS Resident Editor and Assembler to be an extremely useful and powerful program development tool. We are so sure you will be turned on with our Software Package No. 1 that we are practically giving a listing away for a mere $3.00US. Yes, this is a source listing as well as a hexidecimal printout.

The Assembly Language Operating System gives you the ability to write programs in 8080 Assembly Language with labels, expressions and comments. The programs can then be edited by line number, a powerful feature that makes corrections and additions very easy. The program can be named as a file and stored at a user selected memory location while another file is being worked on. Files can be listed by line number using the LIST command before being assembled. The Assembler converts the Assembly Language mnemonic codes and labels to hexidecimal op-codes at any address selected by the user to run at any address (the run address may be different from the location in memory where the program is placed). Assembly can be performed with or without error messages being printed. After assembly the program can be run using the EXECUTE command or dumped onto cassette or paper tape using the DUMP command.

Paper tapes or cassettes of the program listing will not be available to individuals but we have already sent paper tapes to several computer clubs around the country. We suggest you contact one of the clubs if you want a copy of the tape or need assistance. We will be happy to send tape copies to any bona fide "amateur" computer club or society, so if you are a member of such a group, please let us know of your group's existence by sending us a copy of its latest newsletter.

In addition we have a manual describing the use of the ALO System from the ground up. This will include a complete description with examples of every command, instructions on the use of all internal routines by other programs and an overview of efficient file generation and handling.

An even more wonderful version of the ALOS is available in firmware as part of an 8K PROM module. The expanded version allows dynamic Input/Output allocation, file area management by the executive, octal and/or hex data entry, loaders for both 8800 BASIC and Intel Hex Format tapes, and many other capabilities not included in the original Package No. 1. The basic Resident Executive-Editor-Assembler occupies about 4K of the 8K maximum capacity. So why the 8K?? Because we are leaving space for future expansion. The first expansion is a powerful Simulator that adds-on to the basic ALOS package.

SIMULATOR?? Yes, an Interpretive Simulator which runs 8080 programs on the same 8080 that contains the Simulator! Not just traps and breakpoints but simulated I/O, registers, flags, program counter and stack pointer. Any of these can be modified at all times; plus a single step mode that displays all the registers, pointers, flags, etc., after execution of each instruction. This Simulator is the most powerful debugging tool for the 8080 that we know of. Just think, you will hardly ever again have to touch the front panel switches.

Both versions of our ALOS require 2K bytes of RAM for system internal storage and symbol tables. In addition at least 4K more is needed to hold user files, although greater capabilities are achieved with 8K or 12K of user space.

PTCOS!

What is PTCOS you may ask?? It stands for Processor Technology Cassette Operating System and it means a real Operating System program based around our CDS-VIII dual Cassette Data transport System. When operating under this program you have true file handling power to create, delete, edit, relocate, and copy all kinds of files (e.g. BASIC and programs written in BASIC). PTCOS can handle multiple I/O devices using a special type of file and suitable small driving routines. At last an integrated system concept for the 8800 is a reality! PTCOS is devilishly similar in its basic operation to an FDOS and is upward compatible with future software developments from Processor Technology.

PRICE LIST effecitve OCT. 1, 1975

	KIT	ASSEMBLED	DELIVERY
Software package No. 1 Assembly Language Operating System	$3.00	–	3 weeks max.
PTCOS: Processor Technology Cassette Operating System	WRITE		December '75
ALS-8 PROM Firmware module expanded version of SP No. 1	$275.	$325.	3 weeks
SIM-1 PROM Firmware add-on to ALS-8: Simulator section	$95.	$110.	3 weeks

Processor Technology
2465 Fourth Street
Berkeley, Ca. 94710

(415) 549-0857

What is a Character?

by
Manfred Peshka
Peterborough NH 03458

A character is a unit of information used in a communication between a sender and a receiver. Senders and receivers may be either people or machines, or a mix of the two. A character may be represented in different forms: People use mostly graphics, such as the letters of the alphabet, the digits or occasionally the Roman numerals, and the punctuation and special symbols which are so familiar to us. Machines process a set of electric pulses in a period of time which normally represents a character. This time period differs in length for different devices; it is longer for slow devices (terminals, card readers, printers) than for fast devices (tape and disk drives), and is generally the shortest for the computer arithmetic and logical unit.

Parenthetically it should be noted that some machines can recognize graphics, drawings, and even objects (units providing information) in a landscape. The discussion of these machines, however, is reserved to a future article, and their cost is far beyond that of the amateur and hobbyist at the present time.

Symbolic Representation of Alternatives

What is the minimum number of information elements, characters, or basic symbols needed to express an alternative? Probably the most common symbol is the indicator light which tells us that a system is in a specific state as opposed to its "usual" state. Let's consider for a moment the sign "Fire Trucks Entering on Blinking Red Light." This sign indicates the possibility of two specific states: The "usual" state prevails when fire trucks are either on a call or waiting in the garage; in this situation the light is off. The alternative consists of an emergency when the light is blinking to inform people that trucks are about to enter the street, or just have entered and are rushing to the fire. Thereafter the light is again turned off. The light is pulsing for a period of time which normally represents this particular situation or "unit of information," say, about 20 seconds.

The indicator light actually represents the simplest character or basic symbol providing a unit of information. It is binary telling you that a given situation either prevails or not. Similarly, the door bell, the telephone bell, the oil pressure light on your car, etc., are binary symbols. Binary means nothing else but a characteristic, property, or condition of a system in which there are but two alternatives. Besides indicator lights, bells, etc., binary symbols can take on graphic forms such as yes or no, true or false, 1 or 0, to name a few only. For a machine, the form is either the absence or presence of a certain electrical energy level at a period of time of specific duration. While the duration of signaling or "marking" in the case of the oil indicator light may be variable depending on engine rotation, pressure, temperature, etc., it is constant for computing machines. It may be a 1/110th or 1/300th of a second for a slow terminal, or a billionth of a second for a computer central processing unit.

Binary and Ternary Symbol Sets

We have seen that one binary character suffices to indicate two distinct states. On the other hand, an elevator is in one of three states: It is idle, or it is going up, or it is going down. Naturally one binary symbol is not enough to represent three states. Two lights may be used as follows: The left light may signal upward motion when illuminated, and the right light may signal downward motion. No upward or downward motion is indicated when the corresponding light is turned off. Let's represent the two possible states of the indicator lights by the graphics 1 of on and 0 for off. The following three characters then express the three possible states:

00	[oo]	idle
01	[o☀]	down
10	[☀o]	up

Note that a character, that is, the unit of information, is represented by two bits or binary digits. We now have used a two-bit character code to symbolically represent the states of the system consisting of the elevator and its two lights.

An entirely different way to represent three distinct states symbolically is accomplished by increasing the number of basic symbols from two to three. Let's use the graphic 2 to indicate upward movement. Instead of the left and right indicator lights, such conditions may be indicated by a panel displaying the terms idle, down, or up as follows:

0	[IDLE]	neither down nor up
1	[DOWN]	down
2	[UP]	up

This time we used a code consisting of ternary digits to symbolically represent the three states of the elevator and its indicator panel. Ternary means that a characteristic, property, or condition of a system can prevail in one of three alternatives.

Note that the unit of information, or in other words, the character, has been coded in the first case by two binary digits, and in the second instance by one ternary digit. One can conceptualize a character as a distinguishing mark indicating a specific state of a system. Characters are "marks of distinction" which may be represented in different graphic forms which have equivalent value:

Binary	Ternary	Implementation	
00	0	[oo]	[IDLE]
01	1	[o☀]	[DOWN]
10	2	[☀o]	[UP]

The two bit code permits a fourth alternative, namely 11. In actuality, this situation represents a contradiction since the elevator cannot move up and down at the same time. However, this character may be used to signal a defect, such as the elevator being stuck between two floors, or it may simply be out of operation. The ternary code cannot signal this condition unless an additional basic symbol is being used; let's assume that an additional panel indicates a defect when illuminated, and the code representing this situation consists of a binary digit concatenated with a ternary digit as follows:

10	[DEFECT]	[IDLE]
00	[ ]	[IDLE]
01	[ ]	[DOWN]
02	[ ]	[UP]

In this situation, the character or information unit is represented by one binary and one ternary digit. It is a mixed code, principally similar to those found on license plates consisting of letters and decimal digits.

In this situation, two of the six possible characters remain unused, namely 11 and 12. At least, let's hope that they remain unused because 11 would mean that a defective elevator is in downward motion.

Enumerating Alternatives

The number of alternatives which need to be considered in a given system determines the coding requirements. The more alternatives need to be communicated, the more "marks of distinction" are required. We have seen the two basic ways to accomplish this: Increase the number of distinguishing graphics in the character set, or concatenate graphics from the same or from different sets of basic symbols to form strings.

Obviously there is some upper limit to the number of distinguishing marks available to people. Humans have a limit of what they can comfortably memorize in terms of numbers of basic symbols when there is no specific meaning attached to them. Consequently there comes a point when graphics are being concatenated to form symbol strings which represent words. The string 3-D stands for the word which we pronounce 'thrē-'dē and which obviously means "the three-dimensional form or a picture produced in it" (Webster's Seventh New Collegiate Dictionary 1965: page 920). We use the decimal digits 0, 1, 2, . . ., 9 to represent numbers, the letters a, b, c, . . ., z, A, B,

. . ., Z to represent the alphabet for words; special symbols and punctuation marks are concatenated with digits and letters to form even longer strings to represent expressions which inform people about one specific alternative out of, say, a million possibilities. We form mathematical expressions (x^2+x-3, etc.) and word expressions (i.e., sentences) and a combination of the two: "Yesterday it rained in Peterborough for two hours."

The basic unit of information is the basic graphic symbol or character: The space on the paper, the special marks (+ – <, ; etc.), the letters, and the decimal digits, and, which is not immediately obvious, certain functions like the bell on the typewriter which signals the approach of the right margin, the backspace, the margin release, the carrier return, the line feed adjustment, etc. The latter group is called functions or control characters. In the computer and communications field many more functions are encountered than there are on the typewriter. These will be discussed in detail further on.

The number of graphics available for marking one out of many possible states of a system is referred to by the name base. Digits are used to represent numbers; since people generally use ten distinct digits, the number system is called a decimal system. The base of this system is 10. In the previous section the binary number system and the ternary system were used. Their bases are two and three, respectively.

Using any one of these systems, it is possible to mark any number of alternatives. If the number of alternatives exceeds the base (i.e., the number of distinct graphics in the set) one or more additional graphics are used.

Table 1. Equivalence of Selected Graphics.

Binary	Ternary	Octal	Decimal	Hexadecimal
0	0	0	0	0
1	1	1	1	1
10	2	2	2	2
11	10	3	3	3
100	11	4	4	4
101	12	5	5	5
110	20	6	6	6
111	21	7	7	7
1000	22	10	8	8
1001	100	11	9	9
1010	101	12	10	A
1011	102	13	11	B
1100	110	14	12	C
1101	111	15	13	D
1110	112	16	14	E
1111	120	17	15	F
b = 2	3	8	10	16
g = 4	3	2	2	1
a = 16	27	64	100	16

As an example, let's assume that we desired to mark any one of sixteen alternatives. If we used the letters to mark these possibilities, as is often found in term papers and legal documents to mark paragraphs and sections, one graphic for each alternative would suffice. As a matter of fact, out of the 52 available letters only sixteen would be used. Thirty-six graphics would not be used. Two decimal graphics are required to express sixteen options, leaving 84 pairs unused. Three ternary graphics encompass these sixteen possibilities leaving eleven triplets unused. A quadruplet of binary graphics generates exactly sixteen possibilities.

In general, by using 'g' graphics of a set with base 'b', the maximum number of alternatives 'a' is determined by multiplying 'b' with itself for 'g' times, or in other words, $a=b^g$. Table 1 summarizes this rule by enumerating all possible arrangements of binary, ternary, octal (base 8), decimal, and hexadecimal (base 16) graphics for the first sixteen values or alternatives.

To illustrate the rule to calculate the maximum number of alternatives, the hexadecimal system requires only one graphic (g=1) for a maximum of sixteen alternatives (a=16) because its base equals sixteen (b=16). Note, however, that the largest value or number equals fifteen which is represented by the graphic F because enumeration began with the magnitude zero.

The maximum value is always one less than the number 'a' because these systems start counting with zero. Assuming two hexadecimal graphics (g=2), 256 distinct alternatives can be identified ($a=16^2$). The largest value, however, is equal to 255 (a–1) because the first value is zero. The hexadecimal string FF identifies the same magnitude as the decimal string 255 or the bit string 11111111.

It is easy to change from one coding system to another, especially from binary to hexadecimal and back, by means of Table 1. The choice of the hexadecimal graphics A to F was arbitrary and is of great help to people. Machines represent all characters as binary pulses within a given time period. Bit strings, therefore, can become very large and difficult to remember. Imagine the bit string 10001111011100. How much easier it is to

remember the hexadecimal string 23DC instead (you may wish to verify the translation starting with the right four bits). Any other distinct graphics instead of A to F could have been used; for example ! @ # < % >. However, try to remember these in this order, and try to pronounce 23<# instead of the above 23DC.

How to Identify Character Sets

Given the possibility of switching from one representation to another, the question of code identification must be dealt with. Assume the graphic representation 3-D. Is it a word of the English language? Or is it an arithmetic expression? If it is an arithmetic expression, which number system has been employed? Assume another representation such as 11. Which number system has been employed and what magnitude is represented? You may wish to consult Table 1 and calculate the magnitude for each number system.

In order to avoid confusion, graphics other than decimal digits, letters, and the special symbols are identified explicitly. The string 11 therefore means eleven in the decimal number system, and 3-D is part of the English language. If a ternary string was meant, one needs to say so in some unambiguous manner. This can be accomplished through a textual declaration such as "All following digits are ternary digits" or, "The ternary number 11 has a value of 4" where according to our convention the graphic 4 is understood to be a decimal digit.

A different way to identify strings is by appending to the string the base. In the mathematical and computing literature different methods have been employed. In the mathematical literature, this is accomplished by a separate graphic which is appended to the digit string: 11_2 is a binary number with a value of three, while 11_8 is an octal number representing nine, and 11_{16} is a hexadecimal number representing 17. The subscripted graphic represents the base, and it is omitted whenever the base is ten. This convention also avoids the confusion about 3-D. This string is an expression of the English language, whereas 3-D_{16} equals 3-13 or $-A_{16}$ which is a numeric expression resulting in a number.

In the computing literature, different ways have been found to identify bit or hexadecimal strings. These ways depend on the manufacturer and on the computing language employed. In American National Standard (ANS) Fortran, a predominately mathematical language (which is to be distinguished from Basic Fortran), digit strings are recognized as decimal numbers. Bit strings are not allowed, and non-digit strings as used for headlines, table headings, etc., are preceded by one or more digits and the capital letter H; for example, 4H3.14 means the four characters 3.14 which differ in their internal representation from the magnitude 3.14. The constant 4 prior to the H indicates the length of the string; it is four symbols long.

In ALGOL 60 which is an internationally standardized mathematical language, digit strings are recognized as decimal numbers, and character strings for table headings, etc., are enclosed in so-called string brackets as shown in the example: '. . . The wife stated that her husband told her 'our daughter complained 'the teacher is giving me trouble'''. Note that it is possible to have strings within strings, each of which is enclosed by the single quotes pair.

In Programming Language One (PL/I), as devised by IBM, digit strings are recognized as decimal numbers unless they are appended by the letter B. 11B equals 11_2 and has a value of 3. Since the internal representation of binary numbers differs from codes, this language also permits explicit bit and character strings such as '11'B which does not necessarily have a value of 3 but could mean, for example, that the elevator is out of order. Alphanumeric character strings are also permitted and recognized whenever they are enclosed in single quotes: 'THIS IS A "STRING", ISN'T IT?'. Similar distinctions exist also in ALGOL 60 and will be discussed in a future article.

You might have noted that the character constants in Fortran were preceded by the length indicator and an identifying character H. In the systems using quotes or string brackets, the length is determined by the number of positions occupied between the brackets. Many assembler languages combine these two methods. The string is enclosed in quotes, and it is preceded by a single letter indicating the base. B'11' is equal to 11B or '11'B and has a value of 3 when it is used as a number in integer arithmetic. X'11' equals 11_{16} or 17 and is a hexadecimal string.

The distinction between binary numbers and bit strings is a rather fine one and will be discussed in a future article. The computer represents all information as strings of bits and manipulates these strings according to their type in certain groupings of bits. The basic group is called a machine word and consists of one or more bits. These bit groups have an equivalent code value which can be represented graphically in several different ways.

Function Abbreviations

We have discussed earlier various functions of the typewriter. Computer terminals and communications equipment use many more function characters than the common typewriter does. In the various codes, these functions correspond to certain bit strings. The functions are indicated in the code tables on the following pages by abbreviations. Therefore, in Table 2 a dictionary of these abbreviations is presented.

The more frequently encountered terminal function codes (as opposed to transmission functions) are marked with an asterisk.

The Baudot Five-Bit Telegraphy Code

An operator depressing the telegraph key causes current to flow through a wire. The current actuates an electromagnet at the receiving end which produces a "click". The timing between the clicks represents either a dot or a dash, and telegraphers yesterday, and hams today, are skilled in translating these "dots" and "dashes" into graphics.

Transmission speed was mostly dependent on the telegraphers' skills. The term "baud rate" means the frequency at which the dots recurred in a second, with every dash counting twice as long as a dot.

In the automatic teletypewriter the key was replaced by a distributor which sends a fixed number of pulses for each character entered on a keyboard. Latches at the other end actuated a printing device. The term "marking" was used to indicate the flow of current, and the line was "spacing" when the current was off. Marking and spacing can be related to binary digits. In Table 3, a mark is indicated by the bit 1, and a space by the bit 0. In addition to the five bits of the code, a space occurred prior to transmission, and a longer mark (1.5 or 1.42 times the usual mark time) terminated the code. Fig. 1 shows the timing of marks and spaces of the string BYTE:

Fig. 1. The word BYTE in Baudot Code.

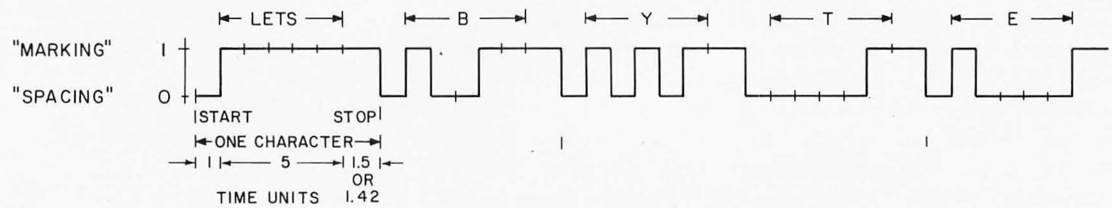


Table 2. Function Abbreviations.

ACK	Affirmative Acknowledgement
BEL, BELL	Bell or other audible signal
BS	Backspace
BYP	By Pass
CAN	Cancel
CC	Cursor Control
CR	Carriage Return
CU 1	Customer Use 1
CU 2	Customer Use 2
CU 3	Customer Use 3
DC Ø	Device Control 0.
DC 1	Device Control 1
DC 2	Device Control 2
DC 3	Device Control 3
DC 4	Device Control 4 (stop)
DEL	Delete
DLE	Data Link Escape
DS	Digit Select
EM	End of Medium
ENQ	Enquiry
EOA	End of Address
EOB	End of Block
EOM	End of Message
EOT	End of Transmission
ERR	Error
ESC	Escape
ETB	End of Transmission Block
ETX	End of Text
FE	Format Effector
FF	Form Feed
FIGS*	Figures Shift
FS	Information File Separator
GS	Information Group Separator
HT	Horizontal Tabulation
IDLE	Null
IFS	Interchange File Separator
IGS	Interchange Group Separator
IL	Idle
IRS	Interchange Record Separator
ITB	Intermediate Text Block
IUS	Interchange Unit Separator
LC	Lower Case
LETS	Letters Shift
LF	Line Feed
NAK	Negative Acknowledgement
NL	New Line
NUL	Null, or all zeros
PF	Punch Off
PN	Punch On
PRE	Prefix
RES	Restore
RS	Record Separator (Reader Stop)
RU	Are you . . . ?
RVI	Reverse Interrupt
S_0–S_7	Separator Information
SI	Shift In
SK	Skip (punched card)
SM	Set Mode
SMM	Start of Manual Message
SO	Shift Off or Shift Out
SOH	Start of Heading
SOM	Start of Message
SOS	Start of Significance
SP	Space
STX	Start of Text
SUB	Start of Special Sequence
SYN	Synchronous Idle
TM	Tape Mark
TTD	Temporary Text Delay
UC	Upper Case
US	Information Unit Separator
VT	Vertical Tabulation
VTAB	Vertical Tabulation
WACK	Wait Before Transmitting Positive Acknowledgement
WRU	Who are you?

Prior to transmission of the letter B, the code LETS must be sent in order to set the receiving equipment into letter shift mode. The reason for this convention is to make it possible to transmit more than 32 symbols with five bits (g=5, b=2, a=32). After all, there are already 26 uppercase letters and ten digits; then there is need for punctuation and special symbols, and function characters to control the printer. Once the operator intends to send a numeric character, the FIGS code is sent prior to the numeric string. In addition to the numeric characters, several other characters were sent in figures shift mode. Depending on the equipment used, various different graphics were assigned to the same bit strings. Table 3 indicates the assignments for four different keyboards; the first column shows the International Telegraph Alphabet No. 2 of the Comite Consultatif International Telegraphique et Telephonique (CCITT); the second column shows the commercial teletype keyboard as used in the United States, the third column presents the fractions keyboard of the American Telephone and Telegraph Company (ATT); the fourth column shows the weather bureau keyboard. All four different keyboards are shown here because used equipment from different sources may be available to you which you might want to modify so that all keycaps correspond to the commercial keyboard.

Table 3. Five-level Baudot Code for Four Selected Keyboards.

BIT CODE						Upper Case			
					Lower Case	CCITT	Commercial	AT & T	Weather
1	1	0	0	0	A	–.	–	–	↑
1	0	0	1	1	B	?	?	5/8	⊕
0	1	1	1	0	C	:	:	1/8	○
1	0	0	1	0	D	Who are you?	$	$	↗
1	0	0	0	0	E	3	3	3	3
1	0	1	1	0	F		!	1/4	→
0	1	0	1	1	G		&	&	↘
0	0	1	0	1	H		#		↓
0	1	1	0	0	I	8	8	8	8
1	1	0	1	0	J	Bell	Bell	'	↙
1	1	1	1	0	K	(	(	1/2	←
0	1	0	0	1	L	)	)	3/4	↖
0	0	1	1	1	M	.	.	.	.
0	0	1	1	0	N	,	,	7/8	◑
0	0	0	1	1	O	9	9	9	9
0	1	1	0	1	P	0	0	0	Ø
1	1	1	0	0	Q	1	1	1	1
0	1	0	1	0	R	4	4	4	4
1	0	1	0	0	S	,	,	Bell	Bell
0	0	0	0	1	T	5	5	5	5
1	1	1	0	0	U	7	7	7	7
0	1	1	1	1	V	=	;	3/8	⦶
1	1	0	0	1	W	2	2	2	2
1	0	1	1	1	X	/	/	/	/
1	0	1	0	1	Y	6	6	6	6
1	0	0	0	1	Z	+	"	"	+
0	0	0	0	0	Blank				—
1	1	1	1	1	Letters shift				↓
1	1	0	1	1	Figures shift				↑
0	0	1	0	0	Space				■
0	0	0	1	0	Carriage return				<
0	1	0	0	0	Line feed				≡

Binary Coded Decimal (BCD) Transmission Code

The term "binary coded decimal" derives from the method of coding decimal digits. The bit string with value 9 is 1001, and the value 10 is expressed by adding an additional four bits, namely, 00010000. The bit string

Table 4. Seven-bit American Standard Code for Information Interchange.

Bits 4	3	2	1	Bits 7, 6, 5 →	000	001	010	011	100	101	110	111
				Hex 1 ↓ / Hex 0 →	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	'	p
0	0	0	1	1	SOH	DC1	|† !	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	¦
1	1	0	1	D	CR	GS	–	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	¬†^	n	~
1	1	1	1	F	SI	US	/	?	O	___	o	DEL

†For IBM 370, the left of the two symbols is generally displayed. See Table 2 for explanation of function abbreviations.

Table 5. Six-bit Binary Coded Decimal Transmission Code.

Bits 3, 4, 5, 6 \ Bits 1, 2	00	01	10	11
0000	SOH	&	–	0
0001	A	J	/	1
0010	B	K	S	2
0011	C	L	T	3
0100	D	M	U	4
0101	E	N	V	5
0110	F	O	W	6
0111	G	P	X	7
1000	H	Q	Y	8
1001	I	R	Z	9
1010	STX	SPACE	ESC	SYN
1011	.	$	)	'
1100	<	*	%	@
1101	BEL	US	ENQ	NAK
1110	SUB	EOT	ETX	EM
1111	ETB	DLE	HT	DEL

01011001 therefore has a value of 59, and 99 is expressed as 10011001. This method differs from the bit coding shown in Table 1.

The binary coded decimal (BCD) transmission code has been widely used by IBM and other manufacturers to transmit uppercase letters, digits, and special symbols in a six-bit code. It is a subset of the USASCII code; however, it is not a national standard. The bit strings are shown in Table 5.

The American Standard Code for Information Interchange (ASCII)

Throughout the decades, many different data transmission codes were developed, and designers today often find good reasons to develop their own codes. The need for standardized transmission codes, however, has increased tremendously because more and more machines dial-up other machines via the public networks. The American Standards Association has standardized a seven bit code for communications. It contains upper and lower-case letters, and a large number of device and transmission control characters. An eighth bit may be added for parity. The term parity implies that the number of bits should add up to an even number (for even parity) or to an odd number for odd parity. The purpose is to check to some degree for a loss of bits during transmission. Assume that a device transmits in even parity; uppercase B consists of two marks and five spaces, therefore, no eighth bit is transmitted; uppercase T consists of three marks and four spaces, and an eighth mark is sent to make the number of marks even. Fig. 2 shows the string BYTE in even parity transmission. The code is shown in Table 4. Bit 1 is transmitted first. You may also want to refer to Table 2 in order to understand the meaning of the abbreviations.

Extended Binary Coded Decimal Interchange Code (EBCDIC)

The Extended Binary Coded Decimal Interchange Code is essentially the previously mentioned Binary Coded Decimal code extended by two bits to form an eight-bit code. A total of 256 codes are possible (b=2, g=8, a=256) and because of its length of eight bits, it is often more easily expressed in hexadecimal notation by means of a string of two hexadecimal digits. Table 6 shows both notations, the bit pattern and the hexadecimal notation. The digit 9, for example, is expressed as the bit string 11111001, or as the hexadecimal string F9.

The code is often used to transmit the eight-bit bytes of computers. It originated about a decade ago when IBM introduced the System 360. The terms "EBCDIC", "byte", and "hexadecimal digits 0, . . ., F" were developed at that time. Today these terms are widely accepted and used by many computer manufacturers. The code is also widely accepted; however, it is not a national standard.

Conclusions

A character is a unit of information which can be represented in various forms, such as in graphic form, or as a bit string. Since bit strings can be rather lengthy and therefore difficult to remember, we discussed the abbreviated representation of the string by means of the hexadecimal graphics. The relationship between the bit string representations of characters and the hexadecimal graphics is independent of the code since it is based on an intrinsic numerical order, namely that of counting from zero by one to infinity.

On the other hand, bit strings may be represented by graphics in an entirely different manner depending on the code used. For that purpose we looked at the predominant five-, six-, seven- and eight-bit codes presently in use. We did not discuss various other but less important codes because of space limitations. Depending on the code utilized, the same graphic represents entirely different bit strings as shown in Table 7.

The first character in the Baudot code is the letters shift. Note the similarity between the last three codes which holds only for uppercase letters and digits.

Fig. 2. The word BYTE in Even-parity USASCII.

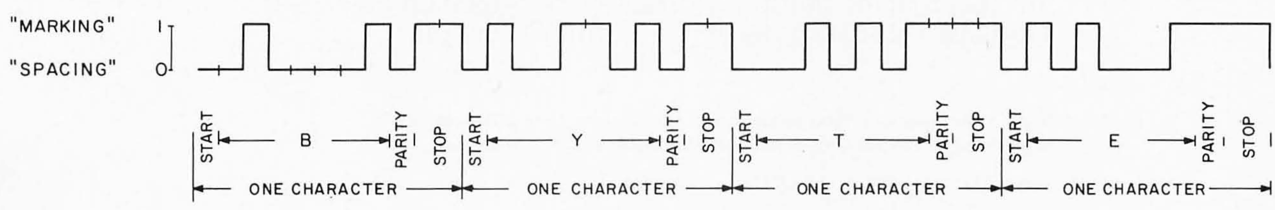


Table 6. Eight-bit Extended Binary Coded Decimal Interchange Code.

Bits 4, 5, 6, 7	Bits 0, 1	00				01				10				11			
	Bits 2, 3	00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
	Hex 0 / Hex 1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NUL	DLE	DS		SP	&	-									0
0001	1	SOH	DC1	SOS				/		a	j			A	J		1
0010	2	STX	DC2	FS	SYN					b	k	s		B	K	S	2
0011	3	ETX	TM							c	l	t		C	L	T	3
0100	4	PF	RES	BYP	PN					d	m	u		D	M	U	4
0101	5	HT	NL	LF	RS					e	n	v		E	N	V	5
0110	6	LC	BS	ETB	UC					f	o	w		F	O	W	6
0111	7	DEL	IL	ESC	EOT					g	p	x		G	P	X	7
1000	8		CAN							h	q	y		H	Q	Y	8
1001	9		EM							i	r	z		I	R	Z	9
1010	A	SMM	CC	SM		¢	!		:								
1011	B	VT	CU1	CU2	CU3	.	$	,	#								
1100	C	FF	IFS		DC4	<	*	%	@								
1101	D	CR	IGS	ENQ	NAK	(	)	_	'								
1110	E	SO	IRS	ACK		+	;	>	=								
1111	F	SI	IUS	BEL	SUB	|	¬	?	"								

Special Graphic Characters

¢	Cent Sign	-	Minus Sign, Hyphen
.	Period, Decimal Point	/	Slash
<	Less-than Sign	,	Comma
(	Left Parenthesis	%	Percent
+	Plus Sign	_	Underscore, Break Character
|	Logical OR	>	Greater-than Sign
&	Ampersand	?	Question Mark
!	Exclamation Point	:	Colon
$	Dollar Sign	#	Number Sign
*	Asterisk	@	At Sign
)	Right Parenthesis	'	Prime, Apostrophe
;	Semicolon	=	Equal Sign
¬	Logical NOT	"	Quotation Mark

See Table 2 for explanation of function abbreviations.

To conclude this tutorial, let me say this in EBCDIC (without start, stop and parity bits):

D5	85	A7	A3	6B
40	A6	85	7D	93
93	40	84	89	A2
83	A4	A2	A2	40
95	A4	94	82	85
99	A2	4B		

■

Table 7. Transmission of the String BYTE in selected codes (excluding start, stop) and parity bits).

11111 10011 10101 00001 10000	Baudot
000010 101000 100011 000101	BCD Transmission Code
0100001 1001101 0010101 1010001	USASCII (see Note 1)
11000010 11101000 11100011 11000101	EBCDIC

Note 1. In memory, the sequence of bits on the IBM 360 and 370 is reversed. The left bit shown becomes the right bit, etc., as shown:

1000010 1011001 1010100 1000101

Special Altair®

MITS-MAS



Christmas Catalog

Lowest Price in the World!

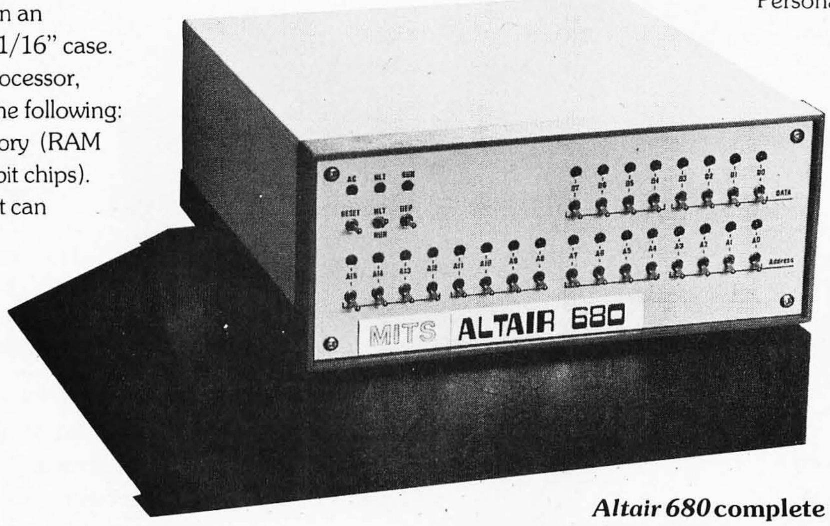
In January of 1975, MITS stunned the computer world with the announcement of the *Altair 8800* Computer that sells for $439 in kit form.

Today MITS is announcing the Altair 680.

The Altair 680, built around the revolutionary new 6800 microprocessor chip, is the lowest priced complete computer on the market. Until December 31, 1975, this computer will be sold in kit form for the amazing introductory price of $293! **(A savings of $52!)**

The Altair 680 comes with power supply, front panel control board, and CPU board inclosed in an 11" wide x 11" deep x 4 11/16" case. In addition to the 6800 processor, the CPU board contains the following:

1. 1024 words of memory (RAM 2102 type 1024 x 1-bit chips).
2. Built-in Interface that can be configured for RS232 or 20 mA Teletype loop or 60 mA Teletype.
3. Provisions for 1024 words of ROM or PROM.



The Altair 680 can be programmed from the front panel switches or it can be connected to a computer terminal (RS232) or a Teletype such as an ASR-33 or surplus five-level Baudott Teletype (under $100).

The Altair 680 can be utilized for many home, commercial or industrial applications or it can be used as a development system for *Altair 680* CPU boards. With a cycle time of 4 microseconds, 16-bit addressing, and the capability of directly addressing 65,000 words of memory and a virtually unlimited number of I/O devices, the *Altair 680* is a very versatile computer!

Altair 680 Software

Software for the *Altair 680* includes a monitor on PROM, assembler, debug, and editor. This software is available to *Altair 680* owners at a nominal cost.

Future software development will be influenced by customer demand and *may include BASIC on ROM.* MITS will sponsor lucrative software contests to encourage the rapid growth of the *Altair 680* software library. Programs in this library will be made available to all *Altair 680* owners at the cost of printing and mailing.

Contact factory for updated information and prices.

Altair Users Group

All *Altair 680* purchasers will receive a free one year membership to the Altair Users Group. This group is the largest of its kind in the world and includes thousands of *Altair 8800* and *680* users.

Members of the Altair Users Group are kept abreast of Altair developments through the monthly publication, ***Computer Notes.***

Altair 680 Documentation

The *Altair 680* kit comes with complete documentation including assembly manual, assembly hints manual, operation manual, and theory manual. Assembled units come with operation and theory manuals. Turnkey model and CPU boards also include documentation.

NOTE: *Altair 680 manuals can be purchased separately. See back page of this catalog for prices.*

Delivery

Personal checks take 2-3 weeks to process while money orders and credit card purchases can be processed in 1-3 days. Delivery should be 30-60 days but this can vary according to order backlog. All orders are handled on a first come, first served basis.

Altair 680 Prices

***Altair 680* complete computer kit $293**
($345 after December 31, 1975)

***Altair 680* assembled and tested $420**

***Altair 680T* turnkey model (complete *Altair 680* except front panel control board) Kit Only $240**
($280 after December 31, 1975)

***Altair 680* CPU board (including pc board, 6800 microprocessor chip, 1024 word memory, 3 way interface and all remaining components except power supply) $180**
($195 after December 31, 1975)

***Altair 680* CPU board assembled and tested $275**

Option I/O socket kit (required when interfacing 680 to external devices) .... $ 29

Option cooling fan (required when expanding 680 internally) .... $ 16
($22 after December 31, 1975)

Option cooling fan installed .... $ 26

PROM kit (256 x 8-bit ultraviolet, erasable 1702 devices) .... $ 42

MITS

"Creative Electronics"

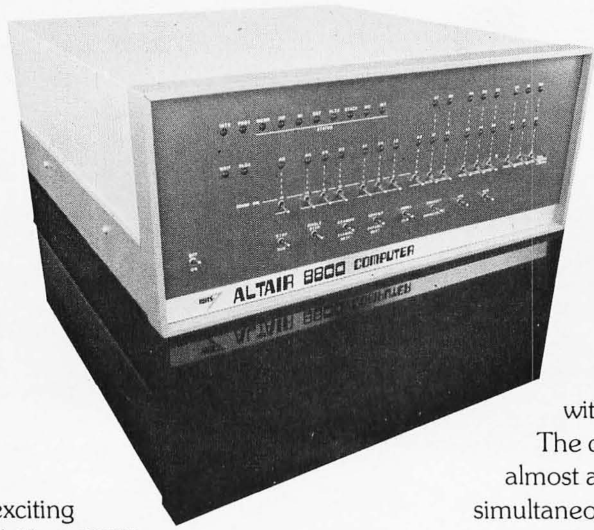
Prices, delivery and specifications subject to change.

"PROJECT BREAKTHROUGH!
World's First Minicomputer Kit To Rival Commercial Models . . . 'Altair 8800'"

headline on cover of *Popular Electronics*, January, 1975

The Altair 8800 from MITS is now one of the most successful computers ever delivered. Thousands of *Altair 8800's* have been sold and are in the field where they are being used for an infinite variety of industrial, business, science and home applications.

The *Altair 8800* is extensively supported by ongoing hardware and software development. *Altair 8800* interface and memory modules and *Altair peripherals* are inexpensively priced, yet among the highest quality in the business. Byte for byte, *Altair 8800* BASIC language software is the most powerful BASIC ever written.



Thanks to the success of the Altair 8800, building and programming computers has become one of the World's most exciting and fastest growing hobbies. Local Altair 8800 Users Clubs have been formed across the United States and in such far away places as England and Japan.

Thanks to clean, efficient design and accurate, easy to understand assembly instructions, the *Altair 8800* is an easy kit to assemble. As an *Altair 8800* kit builder, you will have the satisfaction of successfully building your own computer and you will learn about the internal structure of digital computers.

As the owner of an Altair 8800, you will be backed by the technical expertise of the MITS Customer Service Department. You will receive the latest update information, programming hints, technical advice and general computer information on a monthly basis through a free subscription to *Computer Notes.* You will be in contact with other *Altair 8800* owners through the *Altair Users Group* and you will have access to the extensive *Altair 8800 Software Library.*

No other computer on today's market can offer you as much support as the Altair 8800.

Christmas Special

For a limited time only, you can be the owner of an *Altair 8800* with a 1,024 word memory module for just $68 a month! See back page of this catalog for all the details.

Altair 8800 Features

Built around the most successful (and many say the most powerful) microprocessor chip ever [the Intel 8080], the *Altair 8800* is a variable word length computer with an 8-bit processor, 16-bit addressing and a maximum word size of 24-bits. It has 78 basic machine instructions with variances over 200 instructions. The *Altair 8800* can directly address 256 input and 256 output devices and up to 65,000 words of memory.

Up to 300 peripherals can be interfaced to the *Altair 8800* without any additional buffering. The custom designer can interface almost any number of imaginable devices simultaneously. All *Altair* peripherals are supplied with software handlers to make interfacing easy.

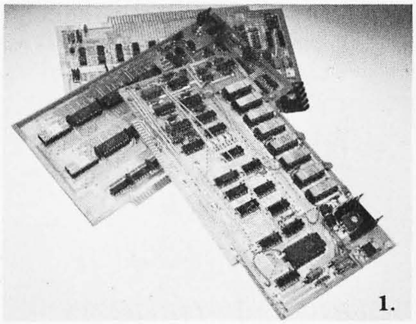
The *Altair 8800* includes the CPU board, front panel control board, power supply (enough to power any additional cards), and expander board (with room for 3 extra interface or memory modules) all inclosed in a handsome, aluminum case complete with sub-panel and dress panel. Up to 16 cards can be added inside the main case.

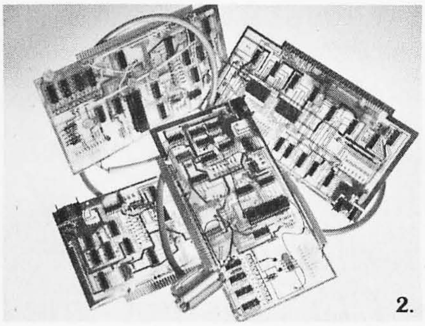
Altair 8800 Prices

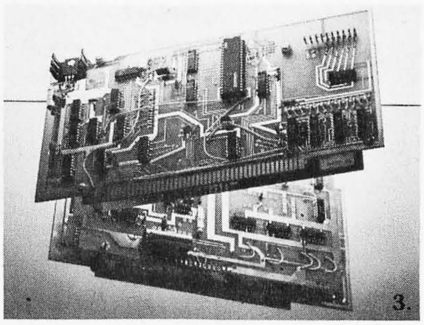
Altair 8800 Computer kit (includes assembly hints, assembly, operator's, and theory manuals)	**$439**
Altair 8800 assembled	**$621**
Expander board (adds 4 slots)	**$ 16 kit**
assembled,	**$ 31**
Cooling fan	**$ 16 kit**
assembled,	**$ 20**

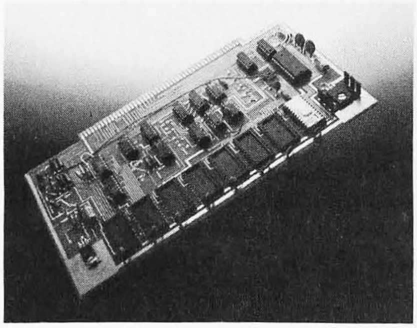
Altair Modules –

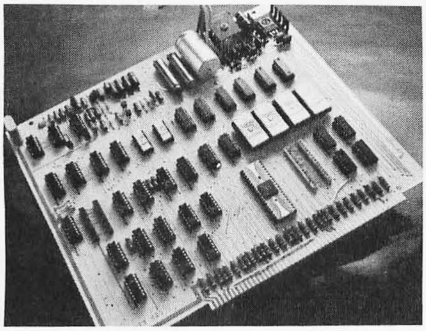
Custom Design Your Computer System to Meet Your Application *(and stay within your budget!)*

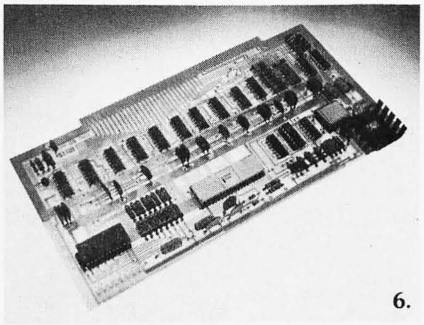












1. Three 8800 memory modules. Three memory modules are now available for the *Altair 8800* and more are in the works. These modules include a 1K (1,024) word static card, a 2K (2,048) word static card, and a 4K (4,096) word dynamic card. Each of these modules is constructed with the finest components available and each contains memory protect features (prevents the computer from accidently writing over programs you want to save). The maximum access time of the static cards is 850 nanoseconds while access time for the dynamic card is 300 nanoseconds.

2. Four 8800 Interface Modules. Interface modules now available for the *Altair 8800* include three serial and one parallell I/O cards. The SIOA serial card is used to connect the *Altair 8800* to CRT's and other computer terminals that have industry standard RS232 asynchronous interconnect lines. It has divider logic to allow for presettable baud rates up to 19,200 baud (5-8 data bits).

The SIOB is the same as the SIOA except that all signals are TTL levels. This card is a general purpose serial interface that can be used to custom interface the *Altair 8800* to a wide variety of devices.

The SIOC is also the same as the other serial cards except it is used to interface the *Altair 8800* to conventional Teletypes and other asynchronous 20 mA current loop terminals. The SIOC is required to interface the *Altair* with ASR-33 Teletypes available from MITS.

The PIO parallel interface card is used for bidirectional transmission of bytes at speeds up to 25,000 bytes/second! It is a full TTL compatible input/output card with necessary handshake flags for conventional parallel interface. Both input and output data have their own 8-bit latch for buffering. Includes necessary logic to allow an adjacent channel to be a control channel. Most commonly used to interface the *Altair 8800* to SWTPC-TVT's or equivalent, custom A/D-D/A interfacing, computer to computer interfacing, and control applications.

3. Audio-Cassette Interface. This best-selling *Altair* module allows you to connect your *Altair 8800* to any tape recorder (medium quality cassette is adequate) for inexpensive mass storage. It works by modulating (changing) digital signals from the computer to audio signals for recording data and by demodulating the audio signal for playback. Consists of a special *Altair* modem board "piggy-backed" on an SIOB board. Requires one slot in the 8800.

4. 8800 PROM module. This PROM memory card is designed to hold up to 2K of PROM. Contact factory for price and other information. *Altair 8800* PROM Programmer to be announced soon.

5. 680 CPU board. The *Altair 680* CPU board is a complete computer on a board (less power supply). In addition to the 6800 CPU available from Motorola and AMI, the *Altair 680* CPU board comes with 1K of RAM memory (2102 type, 1024 x 1-bit chips), built-in I/O that can be configured for RS232 or 20 mA current loop or 60 mA current loop, and provisions for 1K of ROM or PROM. It measures 8¾" x 10¼".

6. 8800 CPU board. The heart of the *Altair 8800* is its CPU board. This double-sided board was designed around the powerful byte oriented, variable word length 8080 processor—a complete central processing unit on a single LSI chip using n-channel, silicon gate MOS technology. The CPU board also contains the *Altair* system clock—a standard TTL oscillator with a 2.000 MHz crystal as the feedback element.

Altair Module Prices:

1K static memory	**$97 kit** and **$139 assembled**
2K static memory	**$145 kit** and **$195 assembled**
4K dynamic memory	**$195 kit** and **$275 assembled**
SIOA interface	**$119 kit** and **$138 assembled**
SIOB and SIOC interface	**$124 kit** and **$146 assembled**
PIO interface	**$92 kit** and **$114 assembled**
Audio-cassette interface	**$128 kit** and **$174 assembled**
680 CPU board kit	**$180 kit**
[$195 after December 31, 1975]	
680 CPU board, assembled	**$275**
8800 CPU board	**$310 kit** and **$360 assembled**

NOTE: *Watch our advertisements for announcement of new Altair 8800 modules and Altair 680 modules.*

BASIC language was chosen for the Altair 8800 because it is the *easiest language to learn* and because it can be used for an *infinite number of applications*. Literally hundreds of thousands of BASIC programs have been written and are in the public domain. These programs include accounting programs, business programs, scientific programs, educational programs, game programs, engineering programs, and much more.

Altair BASIC is an *interactive language*. This means that you get immediate answers and you can use your Altair as a super programmable calculator as well as for writing complicated programs.

Altair BASIC was written as efficiently as possible to allow for the *maximum number of features in the minimum amount of memory*. You can order one of three Altair BASICs: 4K BASIC-designed to run in an Altair 8800 with as little as 4K of memory, 8K BASIC, or EXTENDED BASIC (12K). Each of these BASICs allows you to have *multiple statements per line* (a memory saving feature), and each of them is capable of executing *700 floating point additions* per second!

The 8K BASIC and EXTENDED BASIC have *multi-dimensioned arrays* for both strings and numbers. This is particularly useful for applications requiring lists of names or numbers such as accounting programs, inventory programs, mailing lists, etc.

The 8K BASIC and EXTENDED BASIC also have an OUT and corresponding INP statement that allows you to use your Altair 8800 *control low speed devices* such as drill presses, lathes, stepping motors, model trains, model airplanes, alarms, heating systems, home entertainment systems, etc.

Altair BASIC comes with complete documentation including a copy of *"My Computer Likes Me When I Speak in BASIC"* by Bob Albrecht, a beginner's BASIC text.

Never before has such a powerful BASIC language been marketed at such low prices!

4K BASIC Features

Altair 4K BASIC leaves apporimxately 750 bytes in a 4K Altair for programming which can be increased by deleting the math functions. This powerful BASIC has **16 statements** [IF . . . THEN, GOTO, GOSUB, RETURN, FOR, NEXT, READ, INPUT, END, DATA, LET, DIM, REM, RESTOR, PRINT, and STOP] in addition to 4 commands [LIST, RUN, CLEAR, SCRATCH] and **6 functions** [RND, SQR, SIN, ABS, INT and SGN]. Other features include: *direct execution* of any statement except INPUT; an "@" *symbol* that deletes a whole line and a "←" that deletes the last character; *two-character error code* and line number printed when error occurs; *Control C* which is used to interrupt a program; *maximum line number of 65, 535;* and all results calculated to at least *six decimal digits of precision.*

> ***"I've seen and used other BASICs, but byte-for-byte, Altair is the most powerful BASIC I've seen. I'm particularly impressed with the n-dimensional arrays (and for strings too!), machine level I/O, and machine language 'function' features. The level of your documentation is, for me, though the high point. Sections for those who know nothing and sections for those who know a lot, plus sections that 'normal' people can read and understand."***
>
> **J. Scott Williams**
> **Bellingham, Washington**

8K BASIC Features

Altair 8K BASIC leaves approximately 2K bytes in an 8K Altair for programming which can also be increased by deleting the math functions. This BASIC is the same as the 4K BASIC only with **4 additional statements** [ON . . . GOTO, ON . . . GOSUB, OUT, DEF], **1 additional command** [CONT] and **8 additional functions** [COS, LOG, EXP, TAN, ATN, INP, FRE, POS]. Other additional features include *multi-dimensioned arrays* for both strings and numbers, *AND, OR, NOT* operators that can be used in IF statements or forumlas, *strings* with a maximum length of 255 characters, *string concatenation (A$ = B$)* and the following string functions: *LEN, ASC, CHAR$, RIGHT$, LEFT$, MID$, STR$, and VAL.*

EXTENDED BASIC

Altair EXTENDED BASIC is the same as 8K BASIC with the addition of *double precision arithmetic, PRINT USING* and *disk file I/O*. A minimum of 12K memory is required to support EXTENDED BASIC.

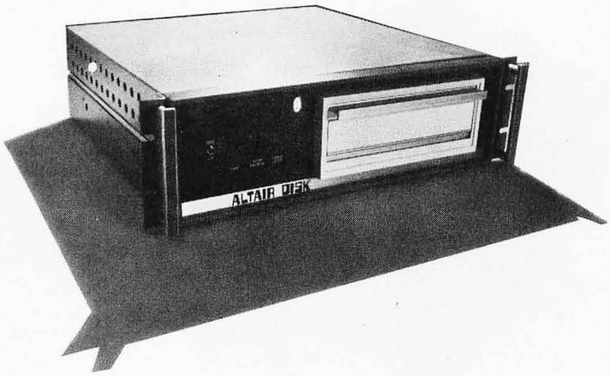
Other Altair 8800 software includes a Disk Operating System, assembler, text editor, and system monitor. Altair users also have access to the *Altair Library*, which contains a large number of useful programs.

SOFTWARE PRICES:

Altair 4K BASIC	**$150**
Purchasers of an Altair 8800, 4K of Altair memory, and an Altair I/O board	**$ 60**
Altair 8K BASIC	**$200**
Purchasers of an Altair 8800, 8K of Altair memory, and an Altair I/O board	**$ 75**
Altair Extended BASIC	**$350**
Purchasers of an Altair 8800, 12K of Altair memory, and an Altair I/O Board	**$150**
Altair PACKAGE ONE (assembler, text editor, system monitor)	**$175**
Purchasers of an Altair 8800, 8K of Altair memory, and an Altair I/O board	**$ 30**
Altair Disk Operating System	**$500**
Purchasers of an Altair 8800, 12K of Altair memory, Altair I/O and Altair Floppy Disk	**$150**

Note: ***When ordering software, specify paper tape or cassette tape.***

Inexpensive, Sophisticated Mass Storage



The *Altair Disk* can store over 300,000 words of information on a floppy disk! It offers the advantage of nonvolatile memory (doesn't "forget" when power is turned off) and fast access to data (3/4 seconds—worst case). The data transfer rate of the *Altair Disk* to and from the computer is a whopping 250,000 bits per second.

The *Altair Disk* includes the disk controller, disk drive, one floppy disk and a software driver. The disk controller, which consists of two cards requiring two slots in the 8800, is capable of controlling up to 16 disk drives. It controls all mechanical functions of the disk, presents the disk status to the computer, and converts serial data to 8-bit parallel words and vice versa for rapid transfer.

The disk drive consists of a Pertec FD400 drive mounted in an *Altair* case including power supply, a buff/multiplexer board, and cooling fan. Its rotational speed is 360 rpm, track to track access time is 10 msec., average time to read or write is 400 msec., and disk life is over 1 million passes per track.

The floppy disk is hard sectored for 32 sectors per track (128 words per sector). There are 77 tracks on each disk. Extra floppy disks are available for $15 from MITS.

A Disk Operating System (software) with complete file structure and utilities for copying, deleting and sorting files is also available.

PRICES:

88-DCDD Altair Disk (includes disk controller, disk drive, one floppy disk and software driver) .... **$1,480 kit**
.... **assembled, $1,980**

88-DISK Altair Disk Drive only .... **$1,180 kit**
.... **assembled, $1,600**

Floppy disk .... **$ 15**

Disk Operating System (if purchased separately) .... **$ 500**
Purchasers of an Altair 8800, 12K of Altair memory, Altair I/O and Altair Disk (88-DCDD) .... **$ 150**

Build Your Own Advanced Terminal!



The Comter II is easily the most advanced computer terminal kit on the market. It has its own internal memory of 256 characters which combines with a highly-readable, soft orange 32 character display to provide ease of operation and information retrieval.

Complete cursor control allows you to move data in and out of the display and operate the Comter II with the versatility of a CRT terminal. **Built-in audio-cassette interface allows you to store unlimited data from the computer and feed that information back into the computer.**

Other features include auto transmit which allows line-by-line transmission of data or program information to the computer from the Comter's memory. The Comter II has a *complete ASCII encoded keyboard* with TTY-33 format plus the addition of special function keys. Its total weight is just 15 lbs.

Flexible power requirements allow you to operate the computer at either 95-125V or 190-250V. The Comter II can be interfaced to any computer with an RS232 serial interface. Requires an SIOA board to be connected to the Altair 8800. No interface required to connect to the Altair 680.

PRICES:

Comter II kit with built-in audio cassette I/O **$780**

Comter II assembled **$920**

High Speed Printing at Low Cost!

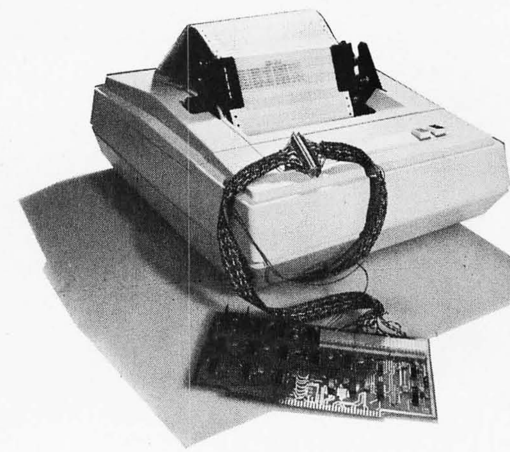
The *Altair 110 Line Printer* is a desktop line printer that produces 80 columns of 5 x 7 dot matrix characters at 100 characters per second x 70 lines per minute. The impact head prints bidirectionally on a 8½" roll paper* using a conventional teletype ribbon. The *Altair Line Printer* will print up to four copies of any item.

Maximum reliability is provided by a mechanism which contains no brakes, clutches, dampers or stepper motors. All control electronics including one-line buffer and self-test circuitry are contained on a single 5" x 15" printed circuit card. The Model 110 was expressly designed for the simplicity, reliability and extremely low cost required by current small-scale data handling systems and terminals.

Vibration and wear are minimized because the print head moves uniformly in both directions and pauses only at the end of each line. Opto-electronic sensing is used to accurately position each dot and permit characters to be printed on the fly.

The *Altair 110 Line Printer* comes with complete control electronics including a printer control card. Requires one slot in the *Altair 8800*.

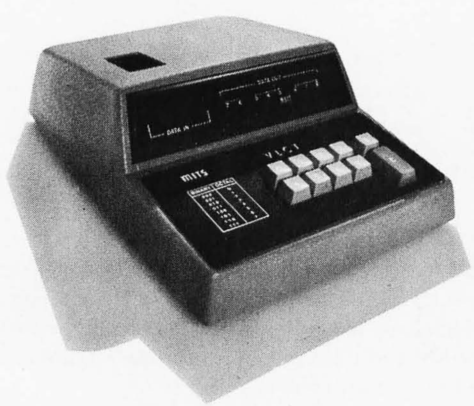
**Pin-fed Optional.*



PRICES:

Altair Line Printer with
controller card .... **$1,750 kit**
.... **assembled, $1,975**

Very Low Cost Terminal



The *Altair VLCT* is ideal for machine language programming. It converts a three digit octal code directly into an 8 digit binary code for transmission to the computer and then displays the binary output from the computer in a 3 digital octal format. This allows you to program in octal which is much easier than programming in binary. In addition, the VLCT is much more convenient to work with than using the front panel switches of the *Altair 8800*.

PRICES:

VLCT .... **$129 kit**
.... **assembled, $169**

NOTE: ***PIO interface module is required to connect VLCT to Altair 8800.***

Teletype — *Most Versatile*



With a built-in paper tape reader and punch and hard copy output, this *ASR-33 Teletype* is perhaps the most versatile of all low cost input/output devices.

The *ASR-33 Teletype* prints 10 characters per second. It is a completely checked-out machine with standard 120 day warranty.

PRICE:

ASR-33 Teletype **$1,500**

NOTE: ***SIOC interface module is required to connect Teletype to Altair 8800.***

Christmas Time Payment Plan

1K Altair for Just $68 a Month!

You can be the owner of an Altair 8800 with a 1,024 word memory module for *just $68 a month.* Each month (for 8 months) you send in your payment and we send you part of an Altair kit until you have the complete system. The advantages of the plan are *NO interest* or financing charge, *GUARANTEED price* based on today's price, and *free, immediate membership to the Altair Users Group* including subscription to Computer Notes.

Our terms are cash with order, BankAmericard, or Master Charge. If you send in an early payment, we will make an early shipment. By the same token, a late payment will result in a late shipment. (After 60 days past due, the balance of the deal is cancelled. All payments must be made within 10 months).

Total $544.00 (Retail price: Altair 8800 $439.00, Memory $97.00, Postage and Handling $8.00—total $544.00)

Altair Users Group Special

Each month the Altair Users Group sponsors a software contest and each month MITS gives away $130 in credit to the winners of this contest. At the end of the year, the author of the overall best program will receive $1000 in credit.

Membership to the Altair Users Group (the largest of its kind in the world) is free to Altair 8800 owners. However, even if you don't own an Altair, *you can be an associate member for one year at the special low price of $10 (regularly $30).*

As a member of the Altair Users Group, you will be kept informed of Altair developments, software contests, and general computer news through the monthly publication, Computer Notes. You will have access to the Altair Software Library and you can communicate to other Altair users throughout the world.

Note: These specials expire on January 30, 1976.

Altair Manuals

Altair 8800 Operators	$ 7.50
Altair 8800 Assembly	$ 9.00
Altair 8800 Theory, Schematics	$10.00
Altair 680 Operators	$ 7.50
Altair 680 Assembly	$ 7.50
Altair 680 Theory, Schematics	$10.00
BASIC Language Documentation	$10.00
Assembler, Monitor, Editor	$ 7.50
8800 4K Memory (includes Assembly, Theory & Schematics)	$ 5.00
8800 2K Memory	$ 5.00
8800 1K Memory	$ 5.00
8800 SIOA	$ 5.00
8800 SIOB	$ 5.00
8800 SIOC	$ 5.00
8800 PIO	$ 5.00
8800 ACR	$ 5.00
COMTER II Operators	$ 6.50
COMTER II Assembly	$10.00
COMTER II Theory, Schematics	$10.00
VLCT (Assembly, Operators, Theory)	$ 5.00
Altair Line Printer Interface	$ 5.00

Documentation Special One

Altair 8800 Operators, Assembly and Theory manuals plus BASIC Language manual (regularly $36.50). *Now just $15.00.*

Documentation Special Two

Altair 680 Operators, Assembly and Theory manuals. Regularly $25. *Now just $14.50.*

Documentation Special Three

BASIC Language manual, Package I (Assembler, Monitor, Editor), and BASIC language beginners text (My Computer Likes Me When I Speak BASIC by Bob Albrecht). Regularly $19.50. *Now just $12.50.*

MITS/6328 Linn NE/Albuquerque, NM 87108
505-265-7553 or 262-1951

Mail this special Altair Coupon Today!

☐ Enclosed is check for $________ ☐ BankAmericard #________ ☐ or Master Charge #________

☐ Altair 8800 ☐ Kit ☐ Assembled ☐ Fan ☐ Christmas Time Plan ☐ Expander Board

☐ Altair 680 ☐ Kit ☐ Assembled ☐ Fan ☐ I/O Sockets

☐ Teletype ☐ Line Printer ☐ Comter II ☐ VLCT ☐ Disk ☐ Disk Drive Only ☐ Disk Controller Only

☐ Memory Module ☐ I/O Module (list on separate sheet)

Postage & Handling: Add $8 for Altair 8800 or Altair 680 and $3 for any cards or peripherals if ordered separately. Teletype and Line Printer shipped by collect freight. Warranty: 90 days on parts for kits and 90 days on parts and labor for assembled units. Prices, specifications, and delivery subject to change.

☐ Documentation Special One ☐ Documentation Special Two ☐ Documentation Special Three

☐ Altair Users Group plus Computer Notes

☐ Please put me on your mailing list

BILL TO:

NAME________

ADDRESS________

CITY________

STATE & ZIP________

COMPANY (IF APPLICABLE)________

SHIP TO:

NAME________

ADDRESS________

CITY________

STATE & ZIP________

☐ Please send Christmas Card to above address announcing gift and anticipated delivery.

MITS/6328 Linn NE/Albuquerque, NM 87108 505-265-7553 or 262-1951

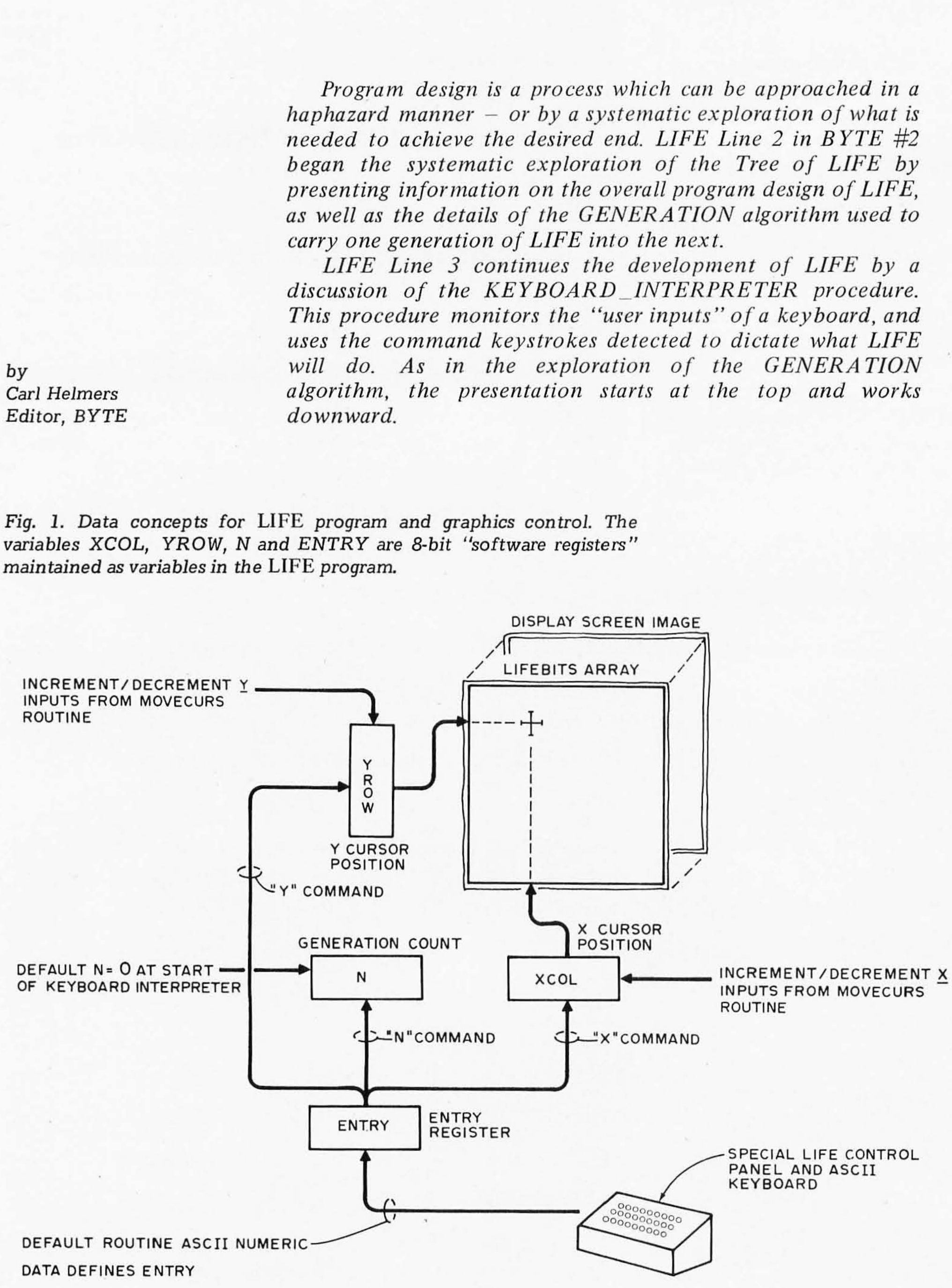
LIFE Line 3

by
Carl Helmers
Editor, BYTE

Program design is a process which can be approached in a haphazard manner – or by a systematic exploration of what is needed to achieve the desired end. LIFE Line 2 in BYTE #2 began the systematic exploration of the Tree of LIFE by presenting information on the overall program design of LIFE, as well as the details of the GENERATION algorithm used to carry one generation of LIFE into the next.

LIFE Line 3 continues the development of LIFE by a discussion of the KEYBOARD_INTERPRETER procedure. This procedure monitors the "user inputs" of a keyboard, and uses the command keystrokes detected to dictate what LIFE will do. As in the exploration of the GENERATION algorithm, the presentation starts at the top and works downward.

Fig. 1. Data concepts for LIFE *program and graphics control. The variables XCOL, YROW, N and ENTRY are 8-bit "software registers" maintained as variables in the* LIFE *program.*



Much of the challenge and fun of the LIFE application is the fact that it is best implemented with some form of interactive graphics. In the partition of the application presented in LIFE Line 2, one of the major pieces of the program is the KEYBOARD_INTERPRETER with its interactive graphics concepts. A good place to start the discussion of the KEYBOARD_INTERPRETER is the *software* block diagram of the interactive graphics system of LIFE.

A Software Block Diagram?

Yes! Strange as it may sound to hardware types, the ebb and flow of data in a program can be depicted in block diagrams. While Fig. 1 looks very much like an ordinary hardware block diagram of some system, it is descriptive of the *plan* of data flow in a program rather than actual wires. Fig. 1 is the programming equivalent in every respect of the hardware block diagram of some dedicated interactive graphics system. By retaining the system in software, LIFE is inherently more flexible than any hard-wired system could be. This block diagram illustrates the potential flow of data in LIFE as controlled through the KEYBOARD_INTERPRETER and its

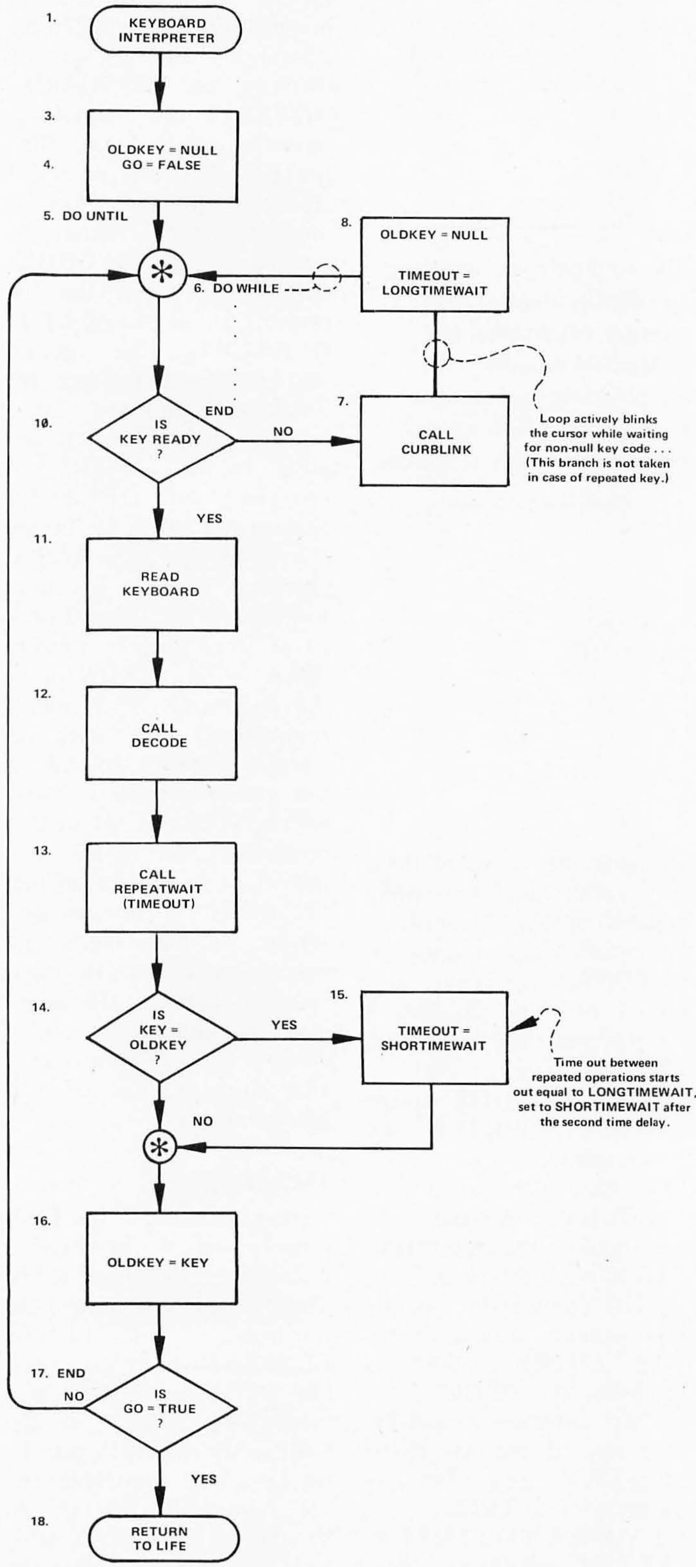


Fig. 2. An overall view of the KEYBOARD_INTERPRETER. *This is a flow chart of the control algorithm for the* LIFE *application's* KEYBOARD_INTERPRETER *routine. Fig. 3 shows the same information in the form of a procedure-oriented language.*

subroutines. Data flows and changes in response to the several input commands defined for the program.

As was pointed out in LIFE Line 1, the fundamental tool of an interactive graphics application is a *cursor* which illustrates where the program thinks attention should be placed. This cursor is flashed on and off on the screen, and can be moved through appropriate commands of the user sent via a keyboard. The cursor concept is implemented in the LIFE program application by means of two "global" variables called XCOL and YROW. These are both 8-bit bytes of data. But since the maximum dimension value in either the X or Y directions of the display is 63 (i.e., 6 bits) only the low order 6 bits have significance for cursor control. At any point in time during the execution of LIFE, the variables YROW and XCOL retain the location of the cursor for KEYBOARD_INTERPRETER's use.

Fig. 1 also shows arrows directed from XCOL and YROW to intersecting dotted lines in the LIFEBITS array. These two numbers together have 12 bits of significance. This is sufficient to uniquely specify one of the 4096 bits in the array using the utility subroutines LGET and LPUT to reference and change LIFEBITS, respectively. These routines are left to a later LIFE Line for their details.

A "ghost copy" of LIFEBITS is also shown in back of the main copy in the drawing to emphasize the following point: Each bit of the internal LIFEBITS array maps directly into a corresponding bit in the refresh memory of the CRT display subsystem. This is an example of a common theme throughout the use and abuse of computer systems: Software systems map into corresponding hardware – and vice versa. This mapping is of course one to one, and is carried out by the DISPLAY subroutine whenever the internal data is changed. As with LGET and LPUT, DISPLAY is left to a future LIFE Line for its details.

What Does it Take to Move the Cursor?

Since the cursor position is maintained by the values of XCOL and YROW, the movement of the cursor is simplicity at its essence: To move the cursor, all you have to do is change the value of XCOL, YROW or both! The interactive graphics portion of KEYBOARD_INTERPRETER has as its primary concern the various ways of

changing the values of these two crucial variables – while possibly leaving a trail of changed data points in LIFEBITS. Fig. 1 illustrates several of these changes –

– To move the cursor up, YROW is incremented.

– To move the cursor down, YROW is decremented.

– To move the cursor left, XCOL is decremented.

– To move the cursor right, XCOL is incremented.

– To completely redefine the column of the cursor, the ENTRY register is transferred to XCOL.

– To completely redefine the row of the cursor, the ENTRY register is transferred to YROW.

KEYBOARD_INTERPRETER performs these actions at the whim of the user via commands entered at the keyboards of the system – with the flashing cursor mark on the screen showing the results.

The simultaneous advantage and disadvantage of the multiple conditional test method of decoding: It is a plodding (but straightforward) approach which squanders memory resources.

The ENTRY Register

In order to provide a means of entering 8-bit integers into the program for control purposes, the software of KEYBOARD_INTERPRETER maintains a numeric input area called ENTRY. Whenever an ASCII character is sent to the program which cannot be decoded by DECODE's COMMAND table, the last resort is to call DEFAULT. In DEFAULT, the recover assumption is to interpret the unknown command as a numeric digit (0 to 9) and push it into ENTRY. A routine in DEFAULT performs a BCD to binary conversion of the ASCII character after it has been trimmed to the range 0 to 9. Later, when the user wants to define XCOL, YROW or N, the commands X, Y and N respectively are used to transfer ENTRY to one of the other registers, after which ENTRY is set to 0 in preparation for re-use. It is important to emphasize that ENTRY is a *binary number.* When decimal digits are entered by the user, the input routines convert the digits into the appropriate binary number and decimally shift the significance of the previous value.

The N Register

In LIFE Line 2, the LIFE program given in Fig. 3 references a variable called N. This N is used to control the number of times GENERATION is called. N, like XCOL and YROW, is a "software register" in the LIFE program which may be set by a user command. The "N" command is what is used to transfer the ENTRY value to N for use in controlling the

Fig. 3. The KEYBOARD_INTERPRETER routine's overall flow, expressed in a procedure-oriented language. Note that the interpretation of the "DO WHILE" differs from a "DO UNTIL" – the former has its test prior to execution of the loop statements, and the latter has its test at the end of the loop. Nesting of the DO groups is indicated by the indentation of lines.

```
KEYBOARD_INTERPRETER:
  PROCEDURE;
  OLDKEY = NULL;
  GO = FALSE;
  DO UNTIL GO = TRUE; /* LOOP UNTIL DONE WITH INPUTS */
    DO WHILE NOTREADY(KEYBOARD) = TRUE;
      CALL CURBLINK;                  /* THIS LITTLE LOOP WAITS */
      OLDKEY = NULL;                  /* FOR A KEYSTROKE AND BLINKS */
      TIMEOUT = LONGWAIT;             /* THE CURSOR ALL THE WHILE */
    END;
    KEY = INPUT(KEYBOARD);            /* WHEN READY, READ KEYBOARD */
    CALL DECODE;                      /* EXECUTE COMMAND */
    CALL REPEATWAIT(TIMEOUT);         /* DON'T LOOK TOO SOON */
    IF KEY = OLDKEY THEN              /* SHORT DELAY AFTER FIRST */
      TIMEOUT=SHORTIMEWAIT;           /* TWO OPERATIONS DONE */
    OLDKEY = KEY;
  END;
CLOSE KEYBOARD_INTERPRETER;
```

Subroutines Referenced by *KEYBOARD_INTERPRETER:*

NOTREADY = a function subroutine (also referenced by *INPUT*) which is used to control an idle loop. It returns *FALSE* as its value if the selected device (in this case, *KEYBOARD*) is ready for input, and it returns *TRUE* as its value otherwise.

CURBLINK = a subroutine which "blinks" the cursor on for a fixed period of time, followed by a fixed period of "off" time. Since it must be called each time a single blink is required, this implements the "active control" feature mentioned in LIFE Line 1.

INPUT = a function subroutine which returns the current input data value for the selected device (in this case, *KEYBOARD*). *INPUT* has its own wait loop referencing *NOTREADY* – which for KEYBOARD_INTERPRETER is redundant, but is not redundant in general.

DECODE = the major subroutine of *KEYBOARD_INTERPRETER.* This routine analyzes *KEY* based upon tables and the previous inputs to the program from the operator. Using this analysis it will select the appropriate subroutine to execute. These "command subroutines" will in turn affect *LIFE* program data and the course of the *LIFE* program's execution.

REPEATWAIT = a subroutine designed to call *CURBLINK* a number of times specified by *TIMEOUT.* This implements a delay between multiple responses to the same key held down continuously.

Data (8-bit bytes) used locally by *KEYBOARD_INTERPRETER:*

OLDKEY = 8-bit value of the last previous keystroke.

NULL = 8-bit value of a null key pattern as read from the keyboard.

TIMEOUT = 8-bit value of the current repeat key time delay.

SHORTIMEWAIT = the timeout parameter used after the first delay in a multiple input of the same key. This specifies the rate of rapid motion of the cursor under manual control.

LONGTIMEWAIT = the value of the timeout parameter used for the first delay following a key entry. A longer wait is required at first to avoid false duplication of keystrokes for heavy-handed players of the game.

Data (8-bit bytes) used by *KEYBOARD_INTERPRETER* and shared with the whole program. See Table II for explanations.

GO, DONE, TRUE, FALSE, KEYBOARD, COMMAND, KEY

extent of the next run. Since this application uses 8-bit data, the limit is 255 generations of LIFE at present.

Figuring Out What the User Said

The KEYBOARD_INTERPRETER routine serves the function of controlling the input of information to these software register and to the LIFEBITS grid. The routine itself is a loop which executes over and over until the user is ready to run the GENERATION algorithm for one or more generations. The KEYBOARD_INTERPRETER terminates for one cycle of LIFE when the user inputs a "G" control command which is interpreted semantically as "GO generate N generations". The flow chart of the KEYBOARD_INTERPRETER logic is illustrated in Fig. 2, with the equivalent procedure-oriented language version shown in Fig. 3 as a detailed reference. In Fig. 2, line numbers are provided for comparison to Fig. 3.

Execution of the KEYBOARD_INTERPRETER begins with some initialization statements. The values of GO and OLDKEY are set at the start of execution (lines 3 and 4). These values will be changed during execution of the KEYBOARD_INTERPRETER based upon input data. OLDKEY is used to detect duplications of keyboard input which occur when a key is held down for continuous operations. After a given KEY is held down continuously for two operations, the repetition goes into a high speed mode with SHORTIMEWAIT controlling the delay between operations. GO is the control variable which is used to govern whether or not the loop is to continue – it is initialized to FALSE and will be changed to TRUE when the "G" user command is decoded.

Programs Are the Willing Servants of the Noble User?

Interaction of programmed computers with human beings is often a waiting game. This waiting game is aptly illustrated in the loop which checks for user input keystrokes at lines 6 to 10 of the KEYBOARD_INTERPRETER routine. The function NOTREADY (KEYBOARD) is a notational convention used to indicate a test for the keyboard ready condition. Like a ready and willing servant, the computer program keeps marking time in this loop until the user – you or I – has given it a character to digest. Two statements are included in this loop for the purpose of coordinating multiple keystroke conditions: Setting OLDKEY = NULL is used to re-establish a null history if the program ever has to wait (it never waits when keys are

Table I. ASCII Command encoding for the LIFE *application. This is an initial specification of the command codes used to control the* KEYBOARD_INTERPRETER *routine's effect. The command table locations go up by three as in Fig. 4. No addresses for the command subroutines are given yet – these will be filled in when the program is compiled for your computer. Command table locations and command characters are given as hexadecimal numbers.*

Command Table Location	Command Character	ASCII Key	Command Subroutine	Meaning of the Command (its "semantics")
00	xx	???	*DEFAULT*	The first table position is the "default" routine position, which is called when no other matching key is found in the table search.
03	47	G	*RUN*	The "run" command which sets a flag called GO in order to end the KEYBOARD_INTERPRETER and have LIFE call the GENERATION routine.
06	49	I	*INITIALIZE*	The "initialize" command to set up the screen with predetermined patterns selected by additional keystrokes.
09	53	S	*SAVELIFE*	The "save" command to dump the current screen content onto a waiting audio cassette or other mass storage device.
0C	52	R	*RESTORELIFE*	The "restore" command to recover a screen pattern previously saved by "S".
0F	58	X	*SETXLOC*	The "set X" command to explicitly set the horizontal cursor location, XCOL.
12	59	Y	*SETYLOC*	The "set Y" command to explicitly set the vertical cursor location, YROW.
15	4E	N	*SETNGEN*	The "set N" command to explicitly set the generation count for subsequent execution with the "G" command.
18	43	C	*CLEARS*	The "clear screen" command to wipe out all data and place the cursor at the center. CLEARS requires confirmation with a second S key stroke to avoid accidental clears.
1B	45	E	*LIFEDONE*	The "done" command is an E followed by an L (for End Life.) The second character confirmation is checked by LIFEDONE.

Note that the ASCII characters 0 to 9 are used to define the "current input" maintained by software in *ENTRY*. *ENTRY* may then be transferred to *N, XCOL,* or *YROW* by the N, X and Y commands respectively.

held down continuously). TIMEOUT = LONGTIMEWAIT re-establishes a longish debounce period between key interpretations following a series of continuous inputs. The program of course thinks that if a key is not ready upon restarting the main loop at line 6 it could not possibly be a repeat. While idling and waiting for your interactive whims, the computer program is not completely devoid of useful work. It calls CURBLINK once each time around the wait loop in order to flash the cursor actively on the screen.

Finally, after some time of unspecified duration, you make up your mind to input a key. This has one major effect upon the program: The next time around the loop at the test of the WHILE condition, a result of FALSE ends the loop. Execution then flows from the DO WHILE (line 6) to line 11 where the KEY is read from the waiting keyboard device by a subroutine called INPUT.

With KEY defined, DECODE is the next item on the agenda. DECODE is one of the major subroutines of KEYBOARD_INTERPRETER, a routine which takes KEY and compares it to a COMMAND table. The result of the COMMAND table search is execution of a "command subroutine" if a match is made, or execution of a DEFAULT routine if no match to KEY is found. Upon return to KEYBOARD_INTERPRETER (all subroutines by nature return to the caller except in very rare cases), the flow of control reaches the REPEATWAIT call using the current value of TIMEOUT.

During normal execution of single isolated commands, the TIMEOUT value is LONGTIMEWAIT – which might be chosen to be from 0.1 to 0.5 seconds. This TIMEOUT sets the minimum time between the first 3 keystrokes of a repeated sequence. But, after two long delays have been executed, the match of OLDKEY = KEY is detected at line 14 and TIMEOUT is changed to SHORTIMEWAIT allowing a speedy repeated motion case. SHORTIMEWAIT might be chosen in the 0.05 to 0.1 second range for rapid motion. The values of these two motion control constants are left unchosen for now, and can be figured out as binary integers to be used in REPEATWAIT when details of the CPU and REPEATWAIT routine are filled in. Note that if fast operation is desired immediately after the second operation of a repeated sequence, then line 13 of Fig. 3 should be moved to a location between lines 15 and 16.

In order to control the repeat logic, the statement OLDKEY=KEY is executed at line 16 so that the last input will be retained for comparison purposes the next time around.

The KEYBOARD_INTERPRETER routine finishes up with the CLOSE statement of LIFE line 18, which stands for the end of the routine and return to its caller. There is one and only one caller of this routine, the LIFE program itself, illustrated in Fig. 3 of LIFE Line #2.

It's All in DECODE of the LIFE Program

When giving the details of the KEYBOARD_INTERPRETER logic, the principle

---

While idling and waiting for your interactive whims, the computer is not completely devoid of useful work. It calls CURBLINK once each time around the wait loop in order to seductively flash its cursor on the screen.

Fig. 4. Decoding by multiple conditional tests. This method of decoding keystrokes and activating routines in software is most efficient when a small number of possible commands is involved.

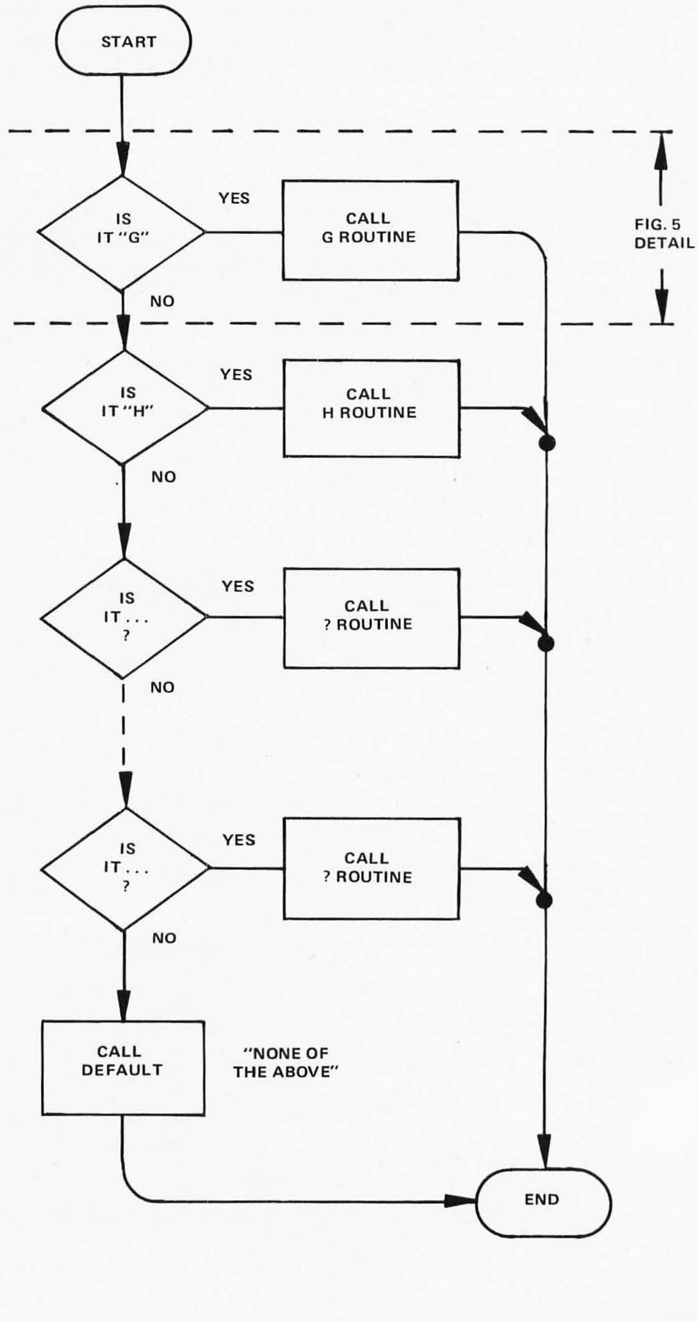


Fig. 5. Typical code for a single conditional test in the scheme of Fig. 4. The example here is using Motorola 6800 system mnemonics. This example assumes accumulator A is set up with the character being decoded.

Bytes	Mnemonic	Comment
2	CMPA # 'G	Compare A to literal
2	BNE * + 4	Branch around JSR and RTS
3	JSR GROUTINE	Call the G subroutine
1	RTS	Return from decoder rather than continue the testing
---		
8 = Total number of bytes per test.		

This is the "generated code" of the following statements in the procedure-oriented language used for LIFE Line examples:

```
IF KEY = 'G' THEN
   DO; /* HAVE MADE A MATCH */
      CALL GROUTINE;
      RETURN; /* FROM DECODE COMPLETELY */
   END;
```

of keeping the program design locally simple results in a CALL DECODE at line 12. Whenever some subroutine is left unspecified except for its inputs (KEY for DECODE) and its outputs (a command subroutine's execution), sooner or later the details must be filled in. In designing a DECODE algorithm to fill in this set of details, there are numerous alternatives. For high order language aficionados, something called a "computed GO TO" (FORTRAN) or "DO CASE" (PL/1 family languages such as XPL or PL/M) would suffice following a table search. However, for this particular application, a slightly lower level approach is justified to conserve memory.

Two major alternatives come to mind as possible ways to map an input KEY value into the execution of a selected subroutine. The simplest (least elaborate) "straightforward" approach is the method of multiple conditional tests. This is illustrated schematically in Fig. 4's flow chart, and in a concrete form in Fig. 5's example of a segment of the typical conditional test. In this approach, each possible command code is tested in turn by the routine. Eventually, all the explicit possibilities will have been exhausted if no match is found. Then, if "none of the above" match the KEY input, a DEFAULT routine is called. The main advantage of this approach is also its disadvantage: It is a plodding and straightforward approach which squanders memory. While the code's intent is obvious, it requires – in the example of Fig. 5 – a total of 8 bytes per test.

There should be a better way – comparisons and branches are repeated in this method. The segment of generated code and its corresponding procedure-oriented language version in Fig. 5 shows four instructions which are repeated over and over but with varying data (the character being compared and the address of the subroutine). Why not put the instructions in only once and tabulate the variable data? There might be a saving of memory if this table driven approach is used instead.

Fig. 6 illustrates the concept of an alternative structure, the "command table," which will result in a lower memory requirement once the number of commands to be tested exceeds some break even point. In this concept, the *changing data* for each test is stored in the table, and the program to go along with it uses a looping technique to scan that table. The changing data for tests comprises:

– The command character. This is the keyboard code which is matched against the actual KEY input.

– The command subroutine. This is the address of the subroutine which will be called if KEY matches the corresponding command character.

The table is organized in 3-byte groups consisting of a command character followed by its subroutine address. Note that on first inspection, this form of DECODE requires only 3 bytes of storage per test versus the 8 bytes in the example of Fig.

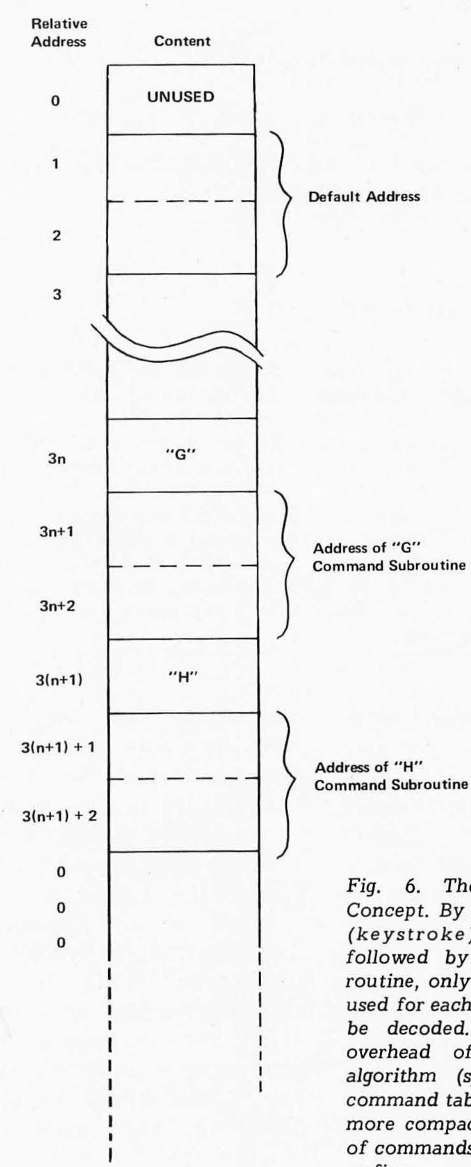


Fig. 6. The Command Table Concept. By storing the character (keystroke) being decoded, followed by the address of its routine, only three bytes need be used for each routine which could be decoded. Allowing for the overhead of a longer decode algorithm (specified once), the command table method will prove more compact when the number of commands get larger than four or five.

Fig. 7. The Command Table DECODE *routine specified in a procedure oriented language.*

```
1   DECODE:
2     PROCEDURE; /* TO FIGURE OUT WHAT USER SAID */
3     /* COME HERE WITH THE KEY TO THE COMMAND */
4     DO FOR I = 3 TO LENGTH(COMMAND) BY 3; /* SCAN TABLE */
5       IF KEY = COMMAND(I) THEN
6         DO; /* WOW!! I GOT A MATCH I GOT A MATCH! */
7           I = I + 1; /* POINT TO ADDRESS ENTRY */
8           CALL CALLX(COMMAND(I));
9           /* NOTATION FOR CALL OF SUBROUTINE, INDEXED */
10          RETURN;
11          /* THIS FORCES EXIT FROM DECODE */
12        END;
13      /* ONLY GET HERE IF NO MATCH IN TABLE */
14      CALL CALLX(COMMAND(1)); /* CALL DEFAULT FROM TABLE */
15    END;
16  CLOSE DECODE;
```

Data (8-bit bytes) used locally by *DECODE:*

I = temporary used for loop control and indexing.

Data (8-bit bytes) used by *DECODE* but shared with the whole program. For details see Table II.

COMMAND, KEY

Subroutines referenced by *DECODE:*

DECODE does not use any "real" subroutines, but does use the following two notational conventions which look like subroutines:

LENGTH(COMMAND) stands for the length (in bytes) of the *COMMAND* table. When you know what it is, you put in the value.

CALLX(X) is used to denote using the two bytes starting at the address X as the address of a subroutine to be called. This is an *indexed* subroutine call effectively. For a Motorola 6800 CPU, this would be performed by an LDX instruction indexed off the *COMMAND* table position, followed by a JSR instruction with the indexed addressing mode.

5. For a 10 command table, this would be a 50 byte saving at first inspection. However, the 50-byte figure does not take into account the longer looping routine required to scan the table and indirectly jump when a match is found. But for 10 commands (the number found in Table I) this 50 byte saving potential goes a long way. I expect the actual DECODE routine of the table driven variety to be considerably less than 50 bytes in length when it is generated for the 6800 system instruction set used as the straw-man in Fig. 5. I'll leave the final conclusion on that to a later LIFE Line.

There is an additional advantage to be obtained from the table driven method. This is an advantage which concerns some of the finer points of programming: The table driven method results in "pure code" in which potentially variable data is completely segregated off in the table. This achieves an often desirable end of separating data from instructions. In the multiple conditional test version, the data of the DECODE is embedded right in the instruction stream, both as the literal value of the character being tested and as the address of the routine being selected. If I want to modify the multiple conditional test version, I must certainly recompile or reassemble the whole routine (a pain in small systems work). In contrast, to modify the table driven version, I only have to alter the table itself, and the variable which specifies the table's length. But this is a minor point in addition to the major memory conservation argument for the table driven approach.

The actual algorithm for DECODE is shown in a procedure-oriented language in Fig. 7. The scan of the table is a DO FOR loop with

Notes on Notation:

Concerning Indentation: The listings of procedures for the LIFE program make use of an indentation convention to help show the structure of the routines. The significance of the indentation is that it shows the opening and closing of various local software constructions and in so doing helps convey the meaning of the program to human readers. Note how the statements from line 7 to line 11 of DECODE in Fig. 7 are indented one level compared to the DO (line 6) and END (line 12) statements. This indentation shows that lines 7 to 11 are part of the DO . . . END construction which is executed if the test on line 5 gives a true result.

The notation "/" followed by arbitary remarks and then a "*/" is the "comments" convention used in these examples. This convention is stolen from the PL/1 family of languages.*

Concerning names of variables: With each procedure specified in LIFE Line, data is separated into two categories: Local data *is used only within the procedure question. Local data may have a name which duplicates names used in other procedures, but is always qualified by its local nature. Thus "I" in* GENERATION *(Fig. 6, LIFE Line 2) is a different data location in memory than the "I" in* DECODE *(Fig. 7, LIFE Line 3).* Data shared with the rest of the program, *which is often called* global data *in programming terminology, is in contrast defined universally for* LIFE. *Global data is summarized for* LIFE *in Table II. Thus whenever* KEY *is referenced (as in* KEYBOARD_INTERPRETER *or in MOVECURS) the same data is intended, since these have been classified as shared or global in the notes accompanying the program listings.*

the index, I, running from 3 (the first entry is reserved for the default) to the length of the table by 3. When a match is found, the 16-bit address in the table is used for an indirect subroutine call (lines 7 and 8). For a Motorola 6800 system, this would be accomplished by an indexed JSR instruction after loading the index register from the table. When the selected command subroutine returns to DECODE (as would any well structured subroutine in the same circumstance), the RETURN statement is executed causing an exit from DECODE and resumption of the KEYBOARD_INTERPRETER at the calling point. If no match is found, the loop eventually runs out and line 14 of Fig. 7 is reached, where the DEFAULT routine is called.

This is shown notationally in a general purpose form with reference to the command table, but in generating the code for the statement of line 14, a simple call to DEFAULT might be substituted. (If the generality of the DECODE routine is to be preserved for possible use with other command tables, this optimization would not be possible.)

What about data for the COMMAND table? Table I provides a preliminary answer to this question by giving a list of command table entries including relative location, the corresponding character code, the ASCII key which invokes the command, the name of the subroutine and a verbal description of the subroutine. This table will be used as the basis for creating a detailed data table when the actual programs of LIFE are generated for a particular computer in a future LIFE Line. For now, Table I serves to list the areas which remain to be covered in the discussion of the KEYBOARD_INTERPRETER and all its subroutines.

LIFE Line 4 will continue the presentation of the KEYBOARD_INTERPRETER portion of the LIFE program. To fill out the remaining portion of the Tree of LIFE, the next installment includes the integration of graphics control commands into the KEYBOARD_INTERPRETER and the first hardware details of LIFE – a simple circuit which combines an ASCII keyboard input with the special purpose controls for an interactive cursor. ■

Does Anyone Know What Happened to Robert T. Wainwright?

This series of articles inadvertantly duplicated the name of Robert T. Wainwright's LIFELINE newsletter, published through 1973. Thanks to Bob Albrecht of People's Computer Co. for sending us his copy of LIFELINE's last issue. Does anyone know where Mr. Wainwright is now (he's no longer at the address given by Charles A. Dunning Jr. in the Letters column), and is LIFELINE still being published?

Table II. Global Data. Data which is shared by an entire program or application is often called "global". The word global is used to indicate the widespread effects of such data in the program's execution. Many procedures will alter and change such data. This table summarizes the global data variables of the LIFE *application as used in procedures given in LIFE Lines #2 and #3.*

COMMAND = the table of commands interpreted by *DECODE,* containing the ASCII codes of command keys and the addresses of the appropriate command subroutine. The format of this table is illustrated in Fig. 6. The information content, in preliminary form, is found in Table I.

DONE = the variable used to control continued execution of the main *LIFE* routine (see LIFE Line #2, Fig. 3).

ENTRY = the entry register used to receive numeric ASCII digits, after weighting the previous value in a BCD fashion. While the entry to *ENTRY* of new digits is done in a BCD manner (multiplying by 10 then adding the digit's value) the content of *ENTRY* is a binary number of 8-bit precision with values 0 to 255 and is thus not itself BCD. (BCD = "binary coded decimal.")

FALSE = the value "0" (00 hex, 000 octal, 00000000 binary). This name is used to indicate the software equivalent of a hardware gate input wired to logical zero.

GO = the flag (value is *TRUE* or *FALSE*) which controls continued execution of *KEYBOARD_INTERPRETER.*

KEY = the 8-bit data area which receives keyboard inputs.

KEYBOARD = the logical unit number of the keyboard I/O device. This is a bit pattern which specifies the device one is talking to.

LIFEBITS = the object of the whole exercise – an array of 64 by 64 *bits* stored as 64 by 8 *bytes.*

N = the variable used to control the number of generations to be evolved by LIFE before returning to *KEYBOARD_INTERPRETER* graphics control.

NCMAX = current maximum column index of live cells.

NCMIN = current minimum column index of live cells.

NCOLMAX = maximum column index of live cells for active area optimization.

NCOLMIN = minimum column index of live cells for active area optimization.

NROWMAX = maximum row index of live cells for active area optimization.

NROWMIN = minimum row index of live cells for active area optimization.

NRMAX = current maximum row index of live cells.

NRMIN = current minimum row index of live cells.

TEMP = 2 by 8 array of bytes containing two 64-bit rows of cells.

THAT = previous line copy index to *TEMP* used in *GENERATION* (see LIFE Line #2, Fig. 6). THAT should always have a value of 1 or 0, opposite of THIS.

THIS = current line copy index to *TEMP* used in *GENERATION* (see LIFE Line #2, Fig. 6). THIS should always have a value of 0 or 1.

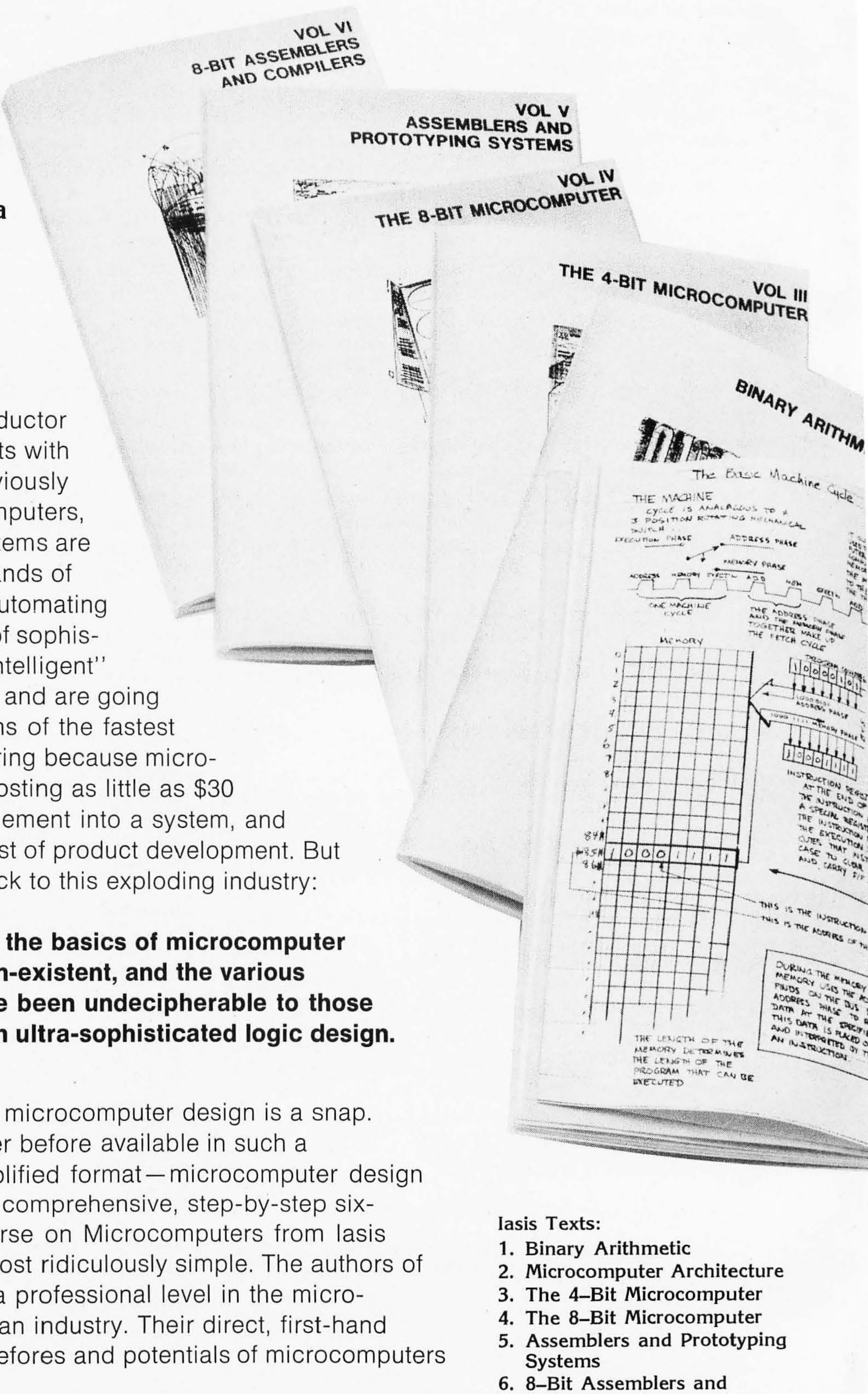
TRUE = the value "255" (FF hex, 377 octal, 11111111 binary). This name is used to indicate the software equivalent of a hardware gate input wired to logical one.

XCOL = the current cursor position in the horizontal (column) direction.

YROW = the current cursor position in the vertical (row) direction.

Microcomputer

A new six-volume programmed learning course from Iasis tells you EVERYTHING about what microcomputers are and how you can design and implement a microprocessor-based system.



Since the transistor was invented, no single electronics innovation has made such an impact as the microcomputer. Powered by tiny semiconductor chips containing computing elements with the same powers and functions previously found only in large scale digital computers, these dedicated microcomputer systems are now being applied to literally thousands of applications. Microcomputers are automating assembly lines, providing the heart of sophisticated electronic games, making "intelligent" computer peripherals even smarter, and are going so far as streamlining the operations of the fastest food chains. This revolution is occuring because microcomputers are very inexpensive—costing as little as $30 in production volume—easy to implement into a system, and significantly reduce the time and cost of product development. But there has been one serious drawback to this exploding industry:

Training materials and courses in the basics of microcomputer technology have been virtually non-existent, and the various published manuals and texts have been undecipherable to those not already intimately familiar with ultra-sophisticated logic design.

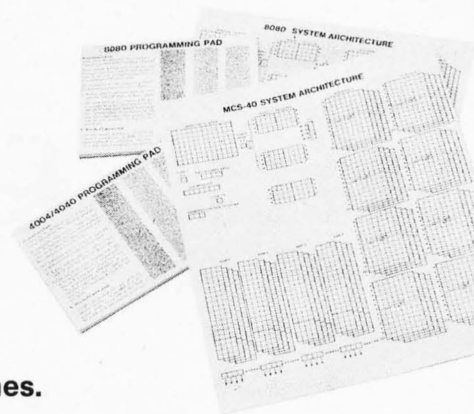
Once a designer has the hang of it, microcomputer design is a snap. But without the fundamentals—never before available in such a readable, understandable and simplified format—microcomputer design has been unbelievably difficult. The comprehensive, step-by-step six-volume Programmed Learning Course on Microcomputers from Iasis makes the unbelievably difficult almost ridiculously simple. The authors of these texts have been involved on a professional level in the microcomputer industry since it became an industry. Their direct, first-hand experience in the whys, hows, wherefores and potentials of microcomputers

Iasis Texts:

1. Binary Arithmetic
2. Microcomputer Architecture
3. The 4–Bit Microcomputer
4. The 8–Bit Microcomputer
5. Assemblers and Prototyping Systems
6. 8–Bit Assemblers and Compilers

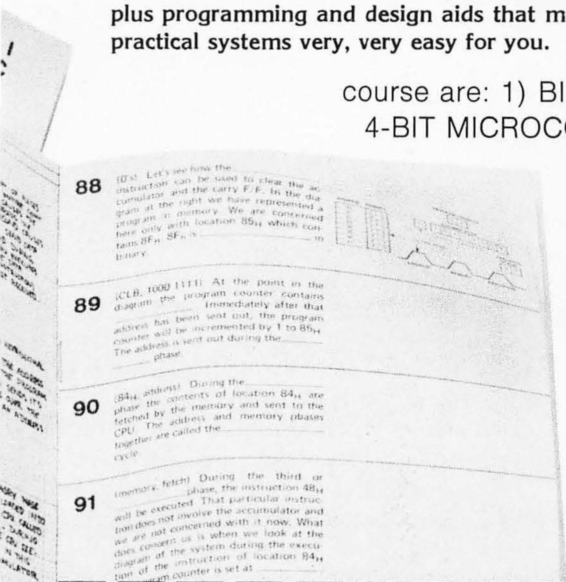
Finally, you can get a comprehensive training course on microcomputers that puts all the hard-to-get information at your fingertips in an easy-to-read, easy-to-understand and even easier-to-implement manner. You can get it here... now.

design is a snap.



have made this six-volume collection the most valuable and meaningful series ever published on microcomputer design. The books combine the most effective methods of programmed instruction with the entire gamut of essential information vital to the designer of a micro-based system. You begin with the ABC's of microcomputers and go through a virtual post-doctoral course... **and the unique, self-testing programmed learning Iasis course enables you to understand and absorb every bit of the information every step of the way through the six volumes.**

The Iasis course gives you more than 700 pages of detailed, illustrated microcomputer information—including more than 1,700 self-tests you use to evaluate your progress—plus programming and design aids that make the design of practical systems very, very easy for you.



Specific details are provided on four of the industry's most versatile microcomputers—the 4004, 4040, 8008 and 8080 from Intel Corporation—but the basic design information will apply to any and all microprocessors. The six volumes you receive with the course are: 1) BINARY ARITHMETIC; 2) MICROCOMPUTER ARCHITECTURE; 3) THE 4-BIT MICROCOMPUTER; 4) THE 8-BIT MICROCOMPUTER; 5) ASSEMBLERS AND PROTOTYPING SYSTEMS; and 6) 8-BIT ASSEMBLERS AND COMPILERS. Plus, this detailed course provides you with two programming pads and two simplified design aids so you may quickly and easily develop both 4-bit and 8-bit microcomputer systems. Use the coupon below to order your course from Iasis, Inc., 110 First St., Suite B Los Altos, California 94022

Special introductory price on this remarkable new course is just $99.50 . . . and if it isn't everything we say it is or even more, return it within 15 days for a full refund!

iasis inc.



Order before Nov. 30, and you'll save a full $25 on the Programmed Learning Course on Microcomputers! In addition, all introductory orders will include a bonus seventh volume, the Microcomputer Applications Handbook!

(After Nov. 30, 1975 price for the complete Iasis course will be $124.50, plus $2.50 for postage and handling.)

Here's my check or money order *(no cash, please). RUSH* my 6-volume Programmed Learning Course on Microcomputers, including the bonus Applications Handbook and programming aids, to the address below. HURRY!

☐ Send one complete course for $99.50 in U. S. funds. My payment is attached. (California residents, please add $5.97 State sales tax.)

☐ Send me information by return mail on quantity discounts.

ALLOW 15 DAYS FOR DELIVERY IN THE U. S. AND 6 WEEKS FOR DELIVERY OUTSIDE THE UNITED STATES.

iasis inc.

Mail today to: Iasis,
110 First St. Suite B
Los Altos, California 94022

You can use your BankAmericard or Master Charge, too!

☐ CHARGE MY ORDER TO THE CREDIT CARD NO. BELOW:

BankAmericard No. ______

Master Charge No. ______
For Master Charge, add 4-digit number immediately above your name on the card. It is ______

HERE'S MY SIGNATURE ______
(Sign here if credit card charge)

Credit card expiration date ______

NAME ______

ADDRESS ______

ORGANIZATION ______ MAIL STOP ______

CITY/STATE/ZIP ______

Flip Flops Exposed

One of the important building blocks in working with transistor-transistor logic is the flip flop. It is important to understand this building block if you desire to use it in projects of your own.

The two most common types of flip flops are known as the JK flip flop and the D flip flop.

JK Flip Flop

The JK flip flop has four or five inputs and one or two outputs. The input pins are labeled J, K, CLOCK, CLEAR and PRESET, and the output pins are labeled Q and $\overline{Q}$ which is often pronounced as "Q bar" or "not Q". A typical block diagram of a JK flip flop is shown in Fig. 1.

The outputs (Q and $\overline{Q}$) can be in one of two states: High (logic 1) or low (logic 0). In general, if the Q output is high then the $\overline{Q}$ output is low, and vice-versa, if the Q output is low then the $\overline{Q}$ output is high. We will often refer to the Q output only since we know that $\overline{Q}$ will be the opposite. So if we say that the output is high it means that Q is high and $\overline{Q}$ is low. However, this is not always so; on some flip flops you may find a Q output only, and, as you will see further below, both outputs may be high or low under specific conditions.

Asynchronous Inputs

Now that we know about the output states, let's discuss the inputs to give us the desired outputs. The PRESET and CLEAR pins are known as asynchronous inputs. Asynchronous means that these inputs do not depend upon the timing derived from the clock pulses.

With almost all flip flops a low PRESET will take Q to high, and a low CLEAR will take the $\overline{Q}$ to high. With both PRESET and CLEAR low both Q and $\overline{Q}$ will be high. This is the only time when Q and $\overline{Q}$ are not opposite from each other. If both PRESET and CLEAR are high, then the control of the output is given to the J, K and CLOCK inputs. The relationship between the asynchronous inputs and the output is shown in the truth table of Fig. 2.

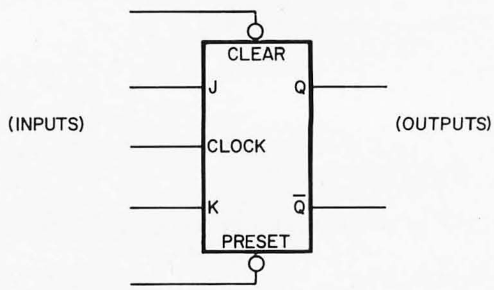


Fig. 1. The JK flip flop block diagram.

by
William E. Browning
516 N. 95th E. Ave.
Tulsa OK 74155

PRESET	CLEAR	Q	$\overline{Q}$
L	L	H	H
L	H	H	L
H	L	L	H
H	H	NO CHANGE	

Fig. 2. Truth table of a JK flip flop responding to preset and clear.

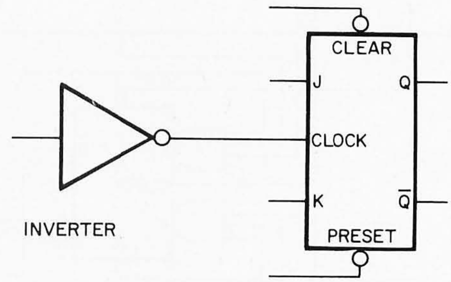


Fig. 4. Inverting the clock input converts rising edge-triggered operation into negative edge-triggered operation and vice versa.

Synchronous Inputs

The J and K inputs are synchronous inputs. Synchronous means that they depend on the CLOCK for operation.

For most JK flip flops a simple set of rules apply to the synchronous inputs, but not all flip flops are standard. There are several variations on the timing when the chip accepts inputs and when the output changes.

The most common input-output relation is known as the rising-edge triggered flip flop. The rising-edge triggered flip flop derives its name from the fact that it changes its output states only when the CLOCK level rises from low to high. Of course, the output states depend on the configuration of the J and K inputs. Fig. 3 illustrates the changes of the Q pin depending on the levels on the J and K pins: When both J and K are low, Q does not change; when both are high, Q changes into its opposite state; with J being low, and K being high, Q assumes a low level, and when J is high, but K is low, Q will take on a high level. Remember, though, that this change can happen only if the CLOCK input rises from low to high, and, as was shown in conjunction with Fig. 2, when both PRESET and CLEAR are high.

The less frequent type of input-output relation is known as the falling-edge triggered flip flop. This flip flop resembles the first, except that the output changes to the condition selected by the J and K inputs when the CLOCK level falls from high to low. Everything else remains the same.

It is easy to change the operation of a falling-edge triggered flip flop to that of a rising-edge triggered flip flop. All that needs to be done is to invert the input to the CLOCK. The inverter shown in Fig. 4 changes a high level to a low level, and vice versa; a rising-edge triggered flip flop can be changed to a falling-edge triggered flip flop, and vice versa, by means of the inverter.

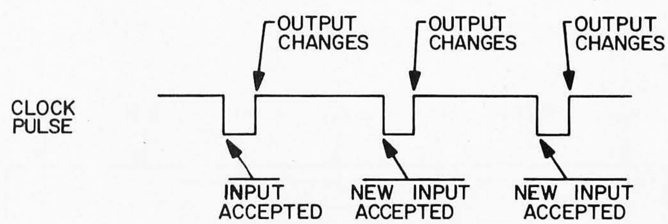


Fig. 5. Timing of a negative clock pulse master/slave flip flop.

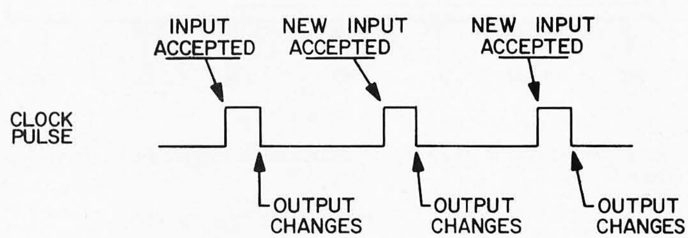


Fig. 6. Timing of a positive clock pulse master/slave flip flop.

t_n		$t_n + 1$
J	K	Q
L	L	Q
L	H	L
H	L	H
H	H	$\overline{Q}$

Fig. 3. Truth table of a JK flip flop for synchronous (clocked) operation.

A third type of input is known as the negative clock pulse master/slave flip flop. With this flip flop the inputs are applied to the J and K pins when CLOCK goes low and change their outputs after the rising edge of the clock pulse (see Fig. 5).

The clock pulses should be made as short as possible and the time between clock pulses as long as possible. This will increase immunity to alternating current noise and accommodate all ripple delay between clock pulses.

A fourth type of input is the positive clock pulse master/slave flip flop. This flip flop accepts inputs when CLOCK goes high and changes output when CLOCK goes low (see Fig. 6). The pulses should be made as short as possible for the above mentioned reasons.

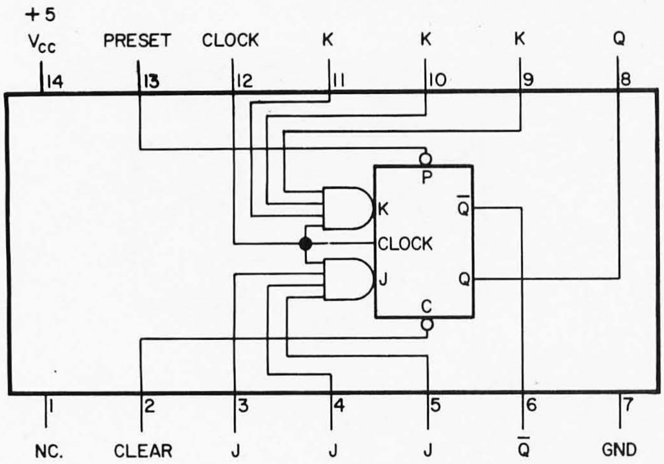


Fig. 7. Pinout of the 7472 IC, a positive clock pulse master/slave flip flop. (Top view)

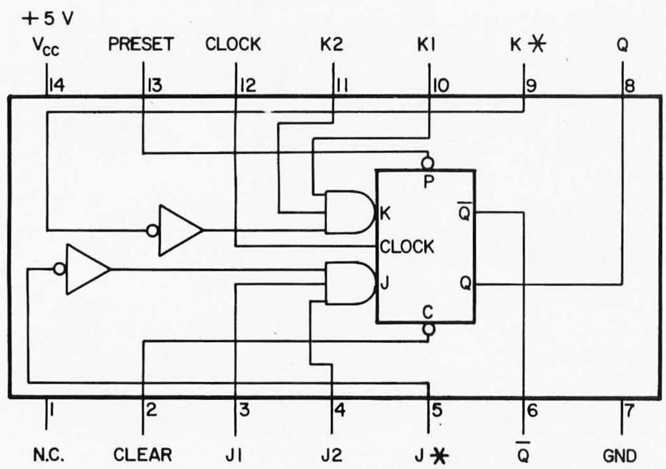


Fig. 8. Pinout of the 7470 IC, a positive edge triggered flip flop. (Top view)

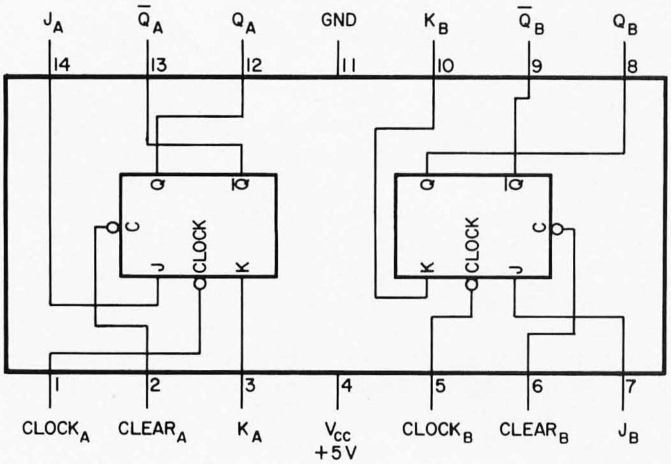


Fig. 9. Pinout of the 7473 IC, a dual positive clock pulse master/slave flip flop. (Top view)

Variations on the Four Flip Flops

One of the confusing aspects with flip flops is that they differ in the logic controlling the J and K inputs. Some have AND/OR logic circuits accepting several distinct inputs, and others have an inverter before the J or K inputs.

If we look at some diagrams of flip flop ICs, their operation will be easier to understand.

The 7472 in Fig. 7 is a single JK flip flop of the positive pulse master/slave type.

The 7472 uses gates on the J and K inputs. In order to get a high level on the J input of the flip flop a high on all three J inputs of the IC (pins 3, 4 and 5) and the clock (pin 12) is required. In a similar manner, a high on the K input of the flip flop is obtained with a high on all three K pins (pins 9, 10 and 11) and the clock of the IC.

If you have only one J input, connect all three J pins together, or connect two of the J inputs to high, and use the remaining input for the data input. The same holds true for the K inputs.

The 7470, shown in Fig. 8, is a single JK flip flop of the positive-edge triggered type.

The 7470 has identical gates and inverters on the J and K inputs. For example, to get a high level on the J input of the flip flop a high on J1 and J2 (pins 3 and 4) and a low on J* (pin 5) must be received. A low level on pin 5 is inverted to a high on the input to the gate.

If you do not need J* or K*, connect them to ground. If you do not need J1, J2, K1 or K2 connect them to high.

The 7473 IC is a dual JK flip flop of the positive pulse master/slave type. Fig. 9 shows that this IC does not use gating on the J and K inputs.

The two flip flops operate separately from each other having only the +5 volt Vcc and ground in common with each other. You may also have noticed that Vcc is on pin 4 and ground on pin 11. This is not the same as on most 7400 ICs which have Vcc on pin 14 and ground on pin 7. So check your connections before you apply power.

There are no PRESET inputs to the chip, in order to allow a 14 pin package.

If you do need the PRESET on a dual JK flip flop, use the 7476 available in a 16 pin package. It is shown in Fig. 10. It has independent CLEAR and PRESET pins, whereas in the 74H78 of Fig. 11 the CLOCK and PRESET pins, respectively, are connected. The 74H78 comes in a 14 pin package.

In many circuits the same clock operates many flip flops and several flip flops are cleared at the same time. The 74H78 is well suited for these needs because it saves on the number of external connections, saves on the size of the IC package, and the number of pins.

D Flip Flop

Let's make one small modification to the JK flip flop: An inverter connects the K input to the J input as shown in Fig. 12.

If J is high then K will be low, and if J is low, K will be high. Since there is only one synchronous input instead of the two, let's call it the data input or D input.

Some flip flops have this modification built into an IC and are referred to as D flip flops. A typical block diagram of a D flip flop is shown in Fig. 13.

The truth table of the D flip flop is shown in Fig. 14. Remember that the PRESET and CLEAR must be high, as discussed in connection with Fig. 2. The truth table for asynchronous inputs applies also to the D flip flop.

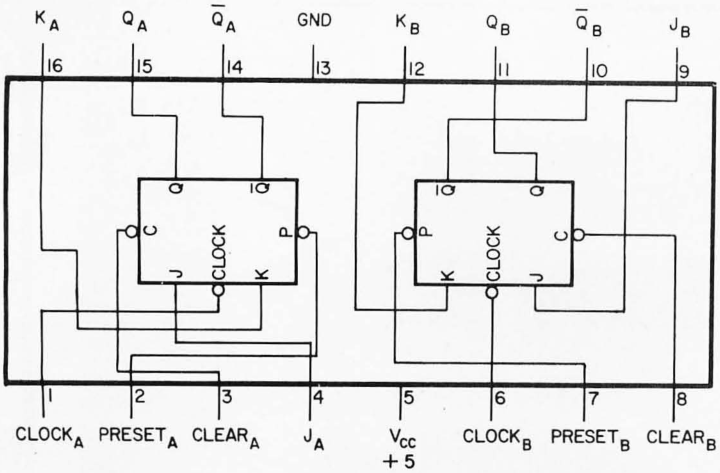


Fig. 10. Pinout of the 7476 IC, a dual flip flop similar to the 7473 but having both PRESET and CLEAR. (Top view)

Several ICs employ the D flip flop. One of these is the 7474 dual D flip flop. Since only one pin is needed for data entry to each flip flop both preset and clear capability can be provided in a 14 pin package. The diagram of the 7474 is shown in Fig. 15.

The 7475 IC uses a D flip flop which is called a latch because the CLEAR and PRESET pins are absent. The reduction in pins has been carried one step further by combining two CLOCK pins each. Therefore it is possible to put four latches on one 16 pin IC; the 7475 quad latch is shown in Fig. 16.

This short introduction to flip flops, latches and integrated circuits should help your understanding of this building block. With a thorough understanding of these circuits, you will be well on your way to designing your own equipment. ■

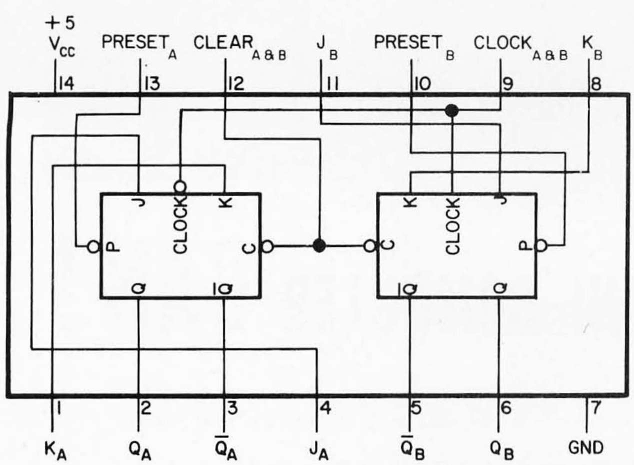


Fig. 11. And still another combination - the pinout of the 74H78 has 14 pins with common CLEAR and CLOCK, and separate PRESET pins.

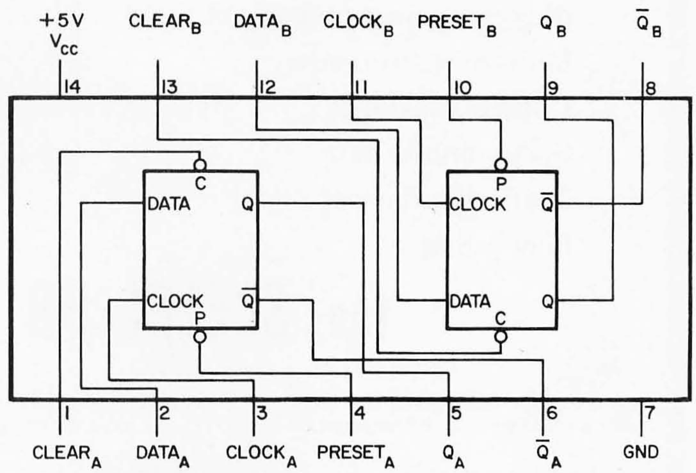


Fig. 15. The pinout of the 7474 dual D flip flop. (Top view)

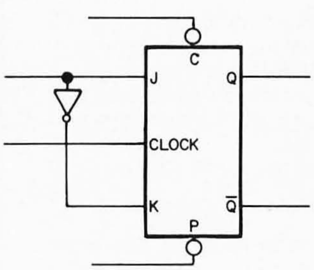


Fig. 12. Making a "D" flip flop out of a "JK" with an inverter. (Top view)

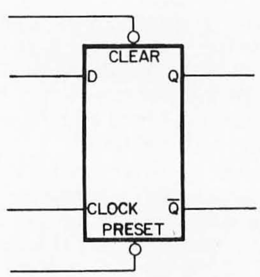


Fig. 13. The block diagram of a D flip flop.

t_n	$t_n + 1$
D	Q $\overline{Q}$
L	L H
H	H L

Fig. 14. Truth table of a D flip flop.

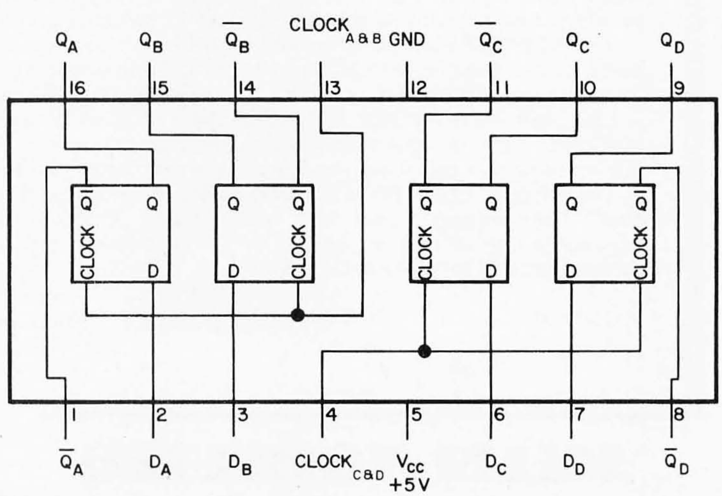
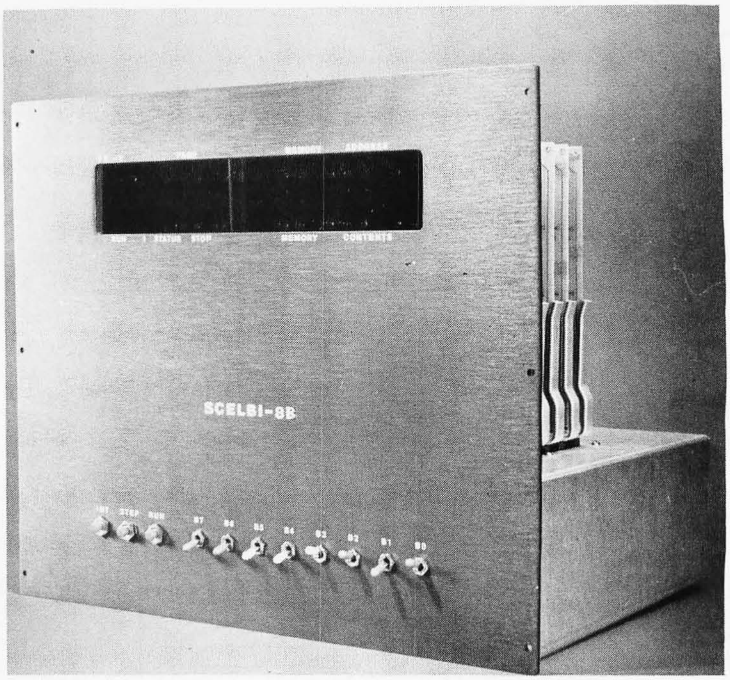


Fig. 16. Pinout of the 7475 latch (or register) circuits. (Top view)

Seasons Greetings

WHAT SINGLE ELECTRONIC MACHINE CAN BE USED TO PERFORM/CONTROL ALL THE FOLLOWING TYPES OF SERVICES?



- Send morse code
- Control repeater stations
- Operate as a calculator
- Receive/send/buffer data between a wide variety of communication devices
- Monitor instruments
- Control machines
- Sort/compile data
- Test other devices
- Play games

the SCELBI-8B MINI-COMPUTER CAN!

SCELBI COMPUTER CONSULTING, INC. - The company that pioneered in producing the small computer for the individual user with the popular SCELBI–8H, now brings you the new SCELBI–8B with increased capability!

Like the former SCELBI–8H, the SCELBI–8B is built around the amazing '8 0 0 8' "CPU-on-a-Chip" which has been revolutionizing the electronics world.

However, the NEW SCELBI–8B offers extended memory capability at reduced cost! It is directly expandable to 16,384 words of RAM/ROM/PROM memory. This increased memory capability now means the user has the potential in a small and compact computer to support compiler type languages, manipulate sizable data bases for business and scientific applications, and support a wide variety of programs including those that take advantage of external mass memory storage devices.

The NEW SCELBI–8B still retains the outstanding features of its predecessor. Decoding logic for 8 Output and 6 Input Ports is built into the basic computer. Plug-in capability for I/O devices is provided on the chassis. A unique, simple to operate console that utilizes just 11 switches on the front panel makes the SCELBI–8B a pleasure to use.

The NEW SCELBI–8B is backed by a line of low cost SCELBI interfaces which currently include: an interface that turns an oscilloscope into an alphanumeric display system, low cost keyboard and TTY interfaces, and an interface that turns a low cost audio tape cassette into a "Mag–Tape" storage and retrieval unit.

Last, but certainly not least, SCELBI has a wide selection of software ready to run on the NEW SCELBI–8B including: Editors, Assemblers, calculating programs, I/O and general utility routines. Additionally, SCELBI produces publications that can show you how to develop your own custom tailored programs.

The NEW SCELBI–8B is available NOW. (We have been delivering since June!) It is available in three forms. Ultra-low cost "Unpopulated" card sets with chassis kits starting at $259.00*. Complete parts kits for a 1,024 word mini-computer as low as $499.00*. An assembled and tested 4,096 word computer is just $849.00*. Interfaces, accessories, and software sold separately.

(*Domestic prices.)

(Prices, specifications and availability subject to change without notice)

Literature available for S.A.S.E.

SCELBI COMPUTER CONSULTING INC.

1322 REAR BOSTON POST ROAD
MILFORD, CONNECTICUT 06460

Read Only Memory Technology

Read Only Memories are a useful element of the hardware of microcomputer systems. This month, Don Lancaster provides information on Read Only Memories – a tutorial article taken from Chapter 3 of his forthcoming book, TV Typewriter Cookbook, *to be published by Howard W. Sams, Indianapolis, Indiana.*

by
Don Lancaster
Synergetics

Low cost and compact memory components are the key to simple and reasonable TV typewriter and microcomputer systems. Today there are many ICs available that will cram thousands or more bits of storage in a single package at costs of a fraction of a cent per bit. The problem is to pick the memory components that are the cheapest, the easiest to use, and the ones with the fewest unpleasant surprises.

What does memory do for us? Well, it remembers. If it remembers extremely well it can be used as a fixed logic element or to store an often-used software routine.

An important example of this is the character generator memory that converts our ASCII code into a group of dot patterns suitable for video use. The same type of memory might be used to change the keyboard switch closures and shift combinations into selected ASCII codes. We also use permanent memory for code conversions such as going from ASCII to SELECTRIC, and permanent memory is usually used to store the program and control commands for an external microprocessor or minicomputer system.

This particular type of memory is called a read only memory, or ROM. Once taught, it does the same thing forever, even if supply power is repeatedly applied and removed. There are several ways this memory type can be taught. One is to use a factory programmed mask at the time of manufacture. Another is to use some field programming technique such as melting internal silicon or metallic fuses. This is handy whenever you are doing a special, low volume code or program, or whenever you

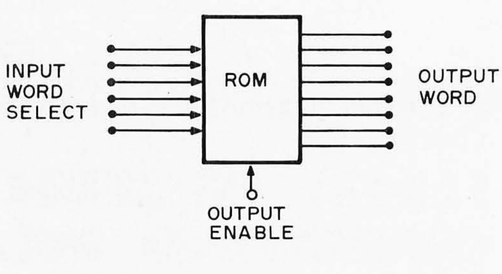


Fig. 1. Read Only Memories (ROMs) are non-volatile and used typically for programming of "firm" microcomputer software, for fixed lookup tables, or for code conversions.

aren't sure what you are doing is really what you want to end up with. Such *programmable read only memories* (PROMs) allow customizing for special purposes. Some of the more expensive programmable read only memories can be bulk-erased through exposure to ultraviolet light or X-rays; these *erasable read only memories* (EROMs) can be re-programmed in the event of an error.

Most read only memories can also be called code converters or table lookup devices, and are usually organized as shown in Fig. 1. Each ROM is a fixed logic block that has several inputs and several outputs. For each *and* every possible input address combination of ones and zeros, some unique combination of output ones and zeros will result. There doesn't have to be any rational relationship between these two code words. Either you or the manufacturer decides what these combinations are going to be at the time the ROM is programmed. A ROM is completely universal; it's inherently set up to provide all possible combinations of input/output word arrangements. When you program your ROM, you limit these all possible combinations to a single specific word exchange that you want.

One popular smaller ROM arrangement is called a 32 x 8 ROM. This means you can program 32 eight bit words. Since 32 words can be represented with binary combinations on five lines, this particular ROM has five input lines and eight output lines. This type of ROM has 256 possible memory locations. At each and every location, we have the option of permanently or semi-permanently placing a one or a zero. This leaves us with 2^{256} possible programs we can teach our ROM, an incredibly large number. The only thing that changes with a particular program is where you put the ones and zeros. All the rest of the circuit stays the same.

ROMs work by decoding each and every possible input state into a one-of-n code and then recombining certain selected combinations of decodings into output words using OR circuits. Which combinations you use picks what the output word is going to be.

There are several ways to program a ROM. It can be done at the factory where metal jumpers are provided or omitted to the tune of holes or no holes in a mask. Factory programming is cheap but must be done with a high volume product that has one internal code that lots of users can agree to use. Dot matrix character generators, some keyboard encoders, trig lookup tables for calculators and so on are typical factory programmed ROMs. Field programmable ROMs are programmed by the user, or by a distributor or someone else who is set up to do programming. A programmable ROM arrives from the factory either all ones or all zeros, depending on the type. You then do something to change the bits you want to suit your code. In one type of ROM, fusable links are melted. These links are made of a metal such as nichrome, or of a semiconductor such as silicon. These techniques are most commonly used on bipolar or TTL-like ROMs. These ROMs are usually fast and relatively small. Another type of programming injects large voltage pulses that avalanche charge storage areas, electret style. This type of programming is used on MOS read only memories. They are usually slower but have more bits available per package.

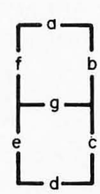
Some premium ROMs are reprogrammable. In one type, you take off an opaque lid and bulk erase the chip with strong ultraviolet light. A second type can have ones or zeros selectively and more or less permanently written into it. Reprogrammable ROMs cost more but can be used over. More important, if you make a programming mistake, you can reuse the same chip, correcting the error later on.

We can also classify ROMs as general purpose devices and dedicated ones. A general purpose ROM can be made into whatever you like and used for just about anything, such as for code conversion, or to store programs for a microcomputer or microprocessor. Dedicated ROMs are usually part of a larger integrated circuit and have very specific uses. Typical examples are in a dot matrix character generator, the word converter in a premium keyboard encoder, and the program storage in many calculator integrated circuits.

Let's see how ROMs work and what devices are available by looking at two important

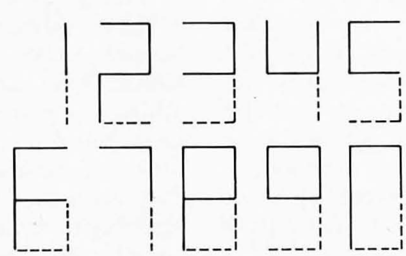
"If you do a PROM design and end up with a ridiculous number of bits, you can almost always go through a rethinking and reduction process that will minimize things a bunch."

Fig. 2. Removing redundant information from the 7-segment calculator display code.



(a) Segment callouts.

(b) Numerals with segments "c" and "d" missing are still identifiable.



"Even with bulk erasable PROMs, a mistake on bit #1874 of a 2k PROM can be enough to ruin your whole day."

Fig. 3. ROM-organized logic to convert 7 segment calculator code to BCD or ASCII code.

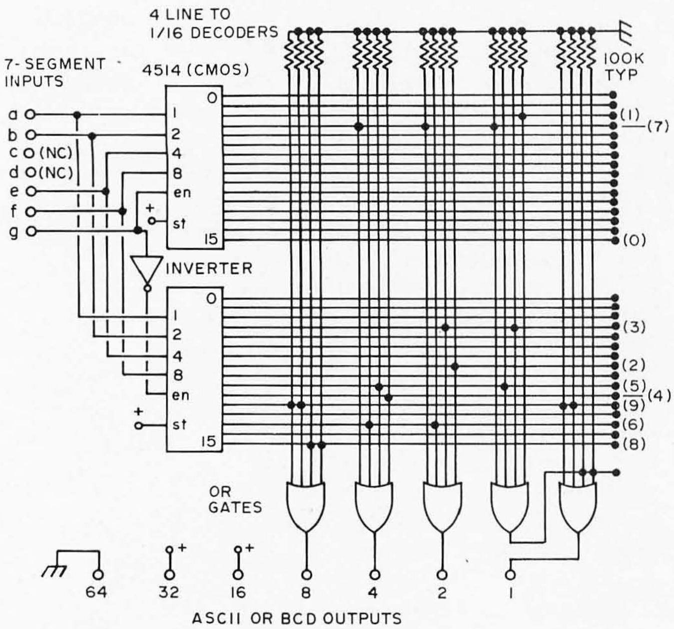
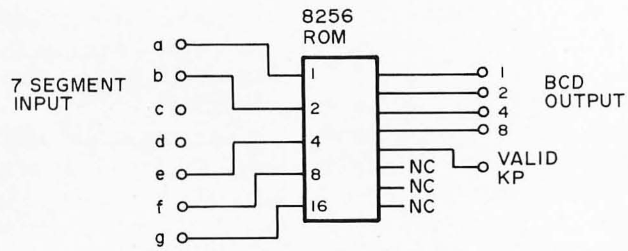


Fig. 4. Single IC 7 segment code converter uses 32 x 8 read only memory.

(a) Circuit using 32 x 8 ROM.



(b) Programming.

INPUT					NUMERAL	OUTPUT				
g	f	e	b	a		kp	8	4	2	1
0	0	0	0	0	–	0	0	0	0	0
0	0	0	0	1	–	0	0	0	0	0
0	0	0	1	0	1	1	0	0	0	1
0	0	0	1	1	7	1	0	1	1	1
0	0	1	0	0	–	0	0	0	0	0
0	0	1	0	1	–	0	0	0	0	0
0	0	1	1	0	–	0	0	0	0	0
0	0	1	1	1	–	0	0	0	0	0
0	1	0	0	0	–	0	0	0	0	0
0	1	0	0	1	–	0	0	0	0	0
0	1	0	1	0	–	0	0	0	0	0
0	1	0	1	1	–	0	0	0	0	0
0	1	1	0	0	–	0	0	0	0	0
0	1	1	0	1	–	0	0	0	0	0
0	1	1	1	0	–	0	0	0	0	0
0	1	1	1	1	0	1	0	0	0	0
1	0	0	0	0	–	0	0	0	0	0
1	0	0	0	1	–	0	0	0	0	0
1	0	0	1	0	–	0	0	0	0	0
1	0	0	1	1	3	1	0	0	1	1
1	0	1	0	0	–	0	0	0	0	0
1	0	1	0	1	–	0	0	0	0	0
1	0	1	1	0	–	0	0	0	0	0
1	0	1	1	1	2	1	0	0	1	0
1	1	0	0	0	–	0	0	0	0	0
1	1	0	0	1	5	1	0	1	0	1
1	1	0	1	0	4	1	0	1	0	0
1	1	0	1	1	9	1	1	0	0	1
1	1	1	0	0	–	0	0	0	0	0
1	1	1	0	1	6	1	0	1	1	0
1	1	1	1	0	–	0	0	0	0	0
1	1	1	1	1	8	1	1	0	0	0

uses for ROMs – a seven bar to ASCII converter that can be used to tie a calculator into a TV typewriter or microcomputer; and an ASCII to SELECTRIC code converter that lets us drive a Selectric typewriter.

Seven Segment Converter

Many calculator chips output only a seven segment code that is not directly compatible with microcomputer software unless it is changed to a Binary Coded Decimal (BCD) or ASCII coding. While several conversion ICs exist, at this writing, they are neither cheap nor readily available. Can we do the job with a read only memory?

At first glance, it would seem that we'd need a ROM with seven inputs or $2^7 = 128$ words minimum. But, with practically every ROM application, a little bit of rethinking can usually drastically cut down the size and cost of the ROM we'll need. For instance, Fig. 2 shows how we can simply ignore the bottom and bottom right segments ("c" and "d") of the segment code and still have ten distinct and identifiable characters. This cuts us down to 32 words, getting us by with five inputs, and one fourth the size of the ROM we started with.

Fig. 3 shows us how we might build our own "pseudo-ROM" using some CMOS gates and decoders. While you would rarely want to go this route, it's useful to look at since it shows us how the real ROMs work inside. You might occasionally use a circuit like this to verify programs and truth tables

since it is easy to change.

Our five input lines (ignoring the redundant c and d segments) are decoded to a one-high-out-of-32 code. For each and every possible input combination, one and only one of the horizontal rails goes to a "1"; the rest stay low. The OR gates on the output re-encode this into a 1-2-4-8 Binary Coded Decimal code. We decide what our OR gates do by where we put the dot connections between the horizontal and vertical rails. To get from BCD to ASCII, we can simply tack a hard wired 011 in front of the BCD word.

While we could dream up a *possibly* simpler "logical minimum" circuit to do the same job, this particular circuit has a unique advantage – if our OR gates are "wide" enough, it will convert ANY five bit input word into ANY four bit output word, with no change in hardware. All that changes is the positions of the dots. This is the beauty of the read only memory – only a single integrated circuit is needed to do an incredible variety of specialized jobs, depending only on how you program it.

Fig. 4 shows how we take a stock 32 x 8, 256 bit Programmable Read Only Memory, or PROM, and do the whole job with one integrated circuit. Since we have extra outputs left over, we can use one for a "valid keypressed" output that can tell the difference between a zero code and no key pressed. The remaining three outputs can be used for detecting a "9" output or for other housekeeping that's handy when you demultiplex the scanning digit outputs of the calculator IC. To program the ROM, the truth table of Fig. 4(b) is entered into the integrated circuit, selectively putting ones and zeros as needed.

MANUFACTURER	PART	BITS	ORG.	TYPE	ERASABLE?
AMERICAN MICRO DEVICES	27S08, 09	256	32x8	BIPOLAR	NO
	27S10, 11	1024	256x4	BIPOLAR	NO
	1702	2048	256x8	MOS	YES
HARRIS SEMICONDUCTOR	1256	256	256x1	BIPOLAR	NO
	8256	256	32x8	BIPOLAR	NO
	0512	512	64x8	BIPOLAR	NO
	1024	1024	256x4	BIPOLAR	NO
	2048	2048	512x4	BIPOLAR	NO
FAIRCHILD	93421	256	32x8	BIPOLAR	NO
	93416,26	1024	256x4	BIPOLAR	NO
	93436,46	2048	512x4	BIPOLAR	NO
INTEL	3601	1024	256x4	BIPOLAR	NO
	3602,22	2048	512x4	BIPOLAR	NO
	1702	2048	256x8	MOS	YES
	3604,24	4096	512x8	BIPOLAR	NO
	2704	4096	512x8	MOS	YES
	2708	8192	1024x8	MOS	YES
INTERSIL	5600,10	256	32x8	BIPOLAR	NO
	5603,23	1024	256x4	BIPOLAR	NO
	5604,24	2048	256x8	BIPOLAR	NO
MONOLITHIC MEMORIES	6330,31	256	32x8	BIPOLAR	NO
	6300,01	1024	256x4	BIPOLAR	NO
	6305,06	2048	512x4	BIPOLAR	NO
	6340,41	4096	512x8	BIPOLAR	NO
NATIONAL	8573,74	1024	256x4	BIPOLAR	NO
	5202,03	2048	256x8	MOS	YES
	5204	4096	512x8	MOS	YES
NITRON	7002	1024	512x2	MOS	YES
	7002	1024	1024x1	MOS	YES
SIGNETICS	8223	256	32x8	BIPOLAR	NO
	82126,29	1024	256x4	BIPOLAR	NO
	82130,31	2048	256x8	BIPOLAR	NO
	82115	4096	512x8	BIPOLAR	NO
TEXAS INSTRUMENTS	74188	256	32x8	BIPOLAR	NO
	74186	512	64x8	BIPOLAR	NO
	74287	1024	256x4	BIPOLAR	NO

Fig. 5. Some commercially available programmable ROMs.

Working With PROMs

Fig. 5 is a listing of some currently available PROMs. Where two numbers are shown, one is usually an open collector output, the other tri-state. At this writing, PROMs cost from $5 upwards, with surplus versions (unused) starting at $3. Bipolar PROMs are based on a TTL technology, usually work off a single +5 volt supply, and are rather fast, typically 50 to 70 nanoseconds access time. MOS PROMs often take two power supplies (+5 and –12 usually) and are slower, typically having a one microsecond access time. MOS PROMs are often cheaper per bit and many MOS types are bulk erasable by exposure to strong ultra-violet light. A few ultra-fast ECL PROMs also exist, but are reserved for special uses and are expensive.

Two good choices for home brew computing are the 32 x 8 bipolar PROM such as the Intersil 5600 or the Signetics 8223; and the 256 x 8 erasable MOS PROM, including the Intel 1702 and its second sources.

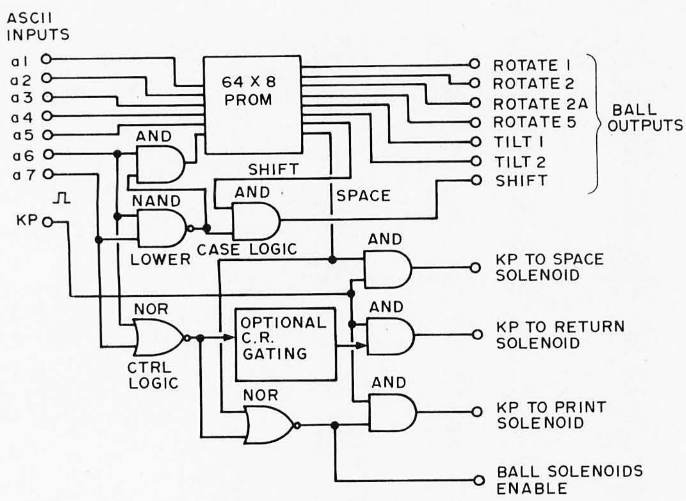
While you can program your own PROM with nothing but a power supply and a meter, the "zero defects" nature of this work and its "up the wall" aspects turn the job into quite a hassle. Even with bulk erasable PROMs, a mistake on bit #1874 of a 2k PROM can be enough to ruin your whole day. Instead of this, you can buy programming services at very low cost from many electronic distributors, as well as from surplus and computer hobby supply houses. Programming machines that simplify the job a bunch are available for several hundred dollars. [Once you have a microcomputer system up and running, it is quite possible to construct an ROM programming peripheral for the purpose of permanently burning in your software. Local computer clubs might consider building the ROM burner peripheral and related software as an attraction of membership . . .cth]

A quarter's worth of gating can cut the size of a ROM in half.

When you design your own PROM circuit, be absolutely sure your truth table is correct before you order any programming. The program service will only guarantee that what you sent in is what you get back, and nothing more. They have no way of second guessing what you really wanted.

If you do a PROM design and end up with a ridiculous number of bits, you can almost always go through a rethinking and reduction process that will minimize things a bunch. Leaving off the two redundant segments of a seven bar code is one obvious example. Other possibilities are to put simple logic outside the PROM, for

Fig. 6. ASCII to SELECTRIC interface using PROM.



often a gate or two can significantly reduce the PROM size. Bypassing control commands around a PROM is one way to do this. Sometimes symmetry and mirror techniques can be used, particularly when working with trig waveforms, music waveshapes, and other data tables that have some sort of symmetry. In PROM microcomputer programs, sneaky programming tricks can often drastically cut the number of steps needed; extensive use of subroutines is one route to this end.

In code converter and table lookup applications you usually address your PROM in a random fashion and you have no way of knowing what is going to be needed next. There are other ways to address ROMs that open up other types of applications. For instance, if you sequentially clock the PROM, changing the address one bit at a time at a constant rate, you can generate an output sinewave or a musical timbre waveform. The clocking rate will select the output frequency, and you can get a symmetrical output by using an up down counter driving the address inputs. Another possibility is to let the PROM's output set the next input address to the PROM, or at least influence it. Some outside latch or storage is needed to prevent an unchecked wild race, but this is easily added.

This particular technique is called microprogramming, and is, of course, the key to calculator and microcomputer operation. Even without a CPU, a PROM plus additional logic can be used as a programmable controller. Loops and branches are easily added by external gating and using extra PROM inputs. Several additional details on PROM and ROM design appeared in the February, 1974, *Radio Electronics*.

A ROM or PROM can be used to change ASCII coded signals into SELECTRIC outputs suitable for the hard copy techniques output if you have a converted Selectric typewriter. While a few ICs are commercially available to do this job (such as the Fairchild 3512 and the National 4230), at this writing, it's much cheaper and simpler to program your own PROM. You can also add custom features of your own, such as converting the ASCII ↑ command into a capital "E" and so on.

Fig. 6 shows us a circuit that only needs 512 bits worth of ROM and a few gates to do the one-way conversion for us. The PROM basically works with the ASCII 6 bit code of upper case alphabets, numbers and punctuation. It converts these ASCII commands into the seven Selectric shift, rotate

Fig. 7. Listing of ASCII inputs (octal) and Selectric outputs for the PROMs in Fig. 6.

ASCII INPUT	CHARACTER	SELECTRIC OUTPUT	ASCII INPUT	CHARACTER	SELECTRIC OUTPUT
100	@	166	040	SPACE	200
101	A	134	041	!	177
102	B	140	042	''	125
103	C	154	043	#	176
104	D	155	044	$	171
105	E	145	045	%	135
106	F	116	046	&	175
107	G	117	047	'	025
110	H	141	050	(	160
111	I	124	051	)	161
112	J	107	052	*	174
113	K	144	053	+	106
114	L	151	054	,	014
115	M	137	055	—	000
116	N	146	056	.	026
117	O	131	057	/	011
120	P	105	060	0	061
121	Q	104	061	1	077
122	R	135	062	2	066
123	S	121	063	3	076
124	T	147	064	4	071
125	U	156	065	5	065
126	V	136	066	6	064
127	W	120	067	7	075
130	X	157	070	8	074
131	Y	101	071	9	060
132	Z	167	072	:	115
133	[	111	073	;	015
134	\	164	074	<	027
135	]	111	075	=	006
136	∧	145	076	>	127
137	—	100	077	?	111

ASCII 6 5 4	ASCII 3 2 1		SELECTRIC SP SH	SELECTRIC T2 T1 R5	SELECTRIC R2A R2 R1	BITS	READING
0 1 0	1 0 1	U	0 1	1 0 1	1 1 0	BINARY	
2	5	U	1	5	6	OCTAL	

and tilt ball commands. The program appears in Fig. 7. We've shown it in octal coding to make it more compact.

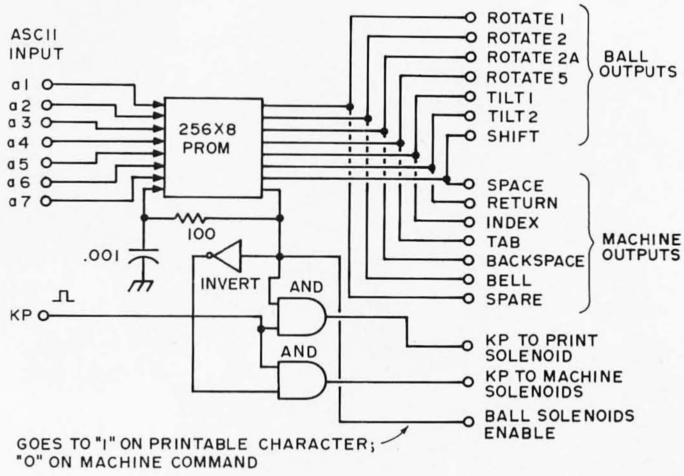
Most of the characters are directly converted from ASCII to their SELECTRIC equivalents. ASCII < and > become the Selectric 1/2 and 1/4 respectively, the ASCII \ becomes the Selectric ¢, and opening and closing brackets are disallowed and produce question marks. ASCII ↑ becomes a capital "E" to indicate exponentiation, particularly when using the BASIC language on a microcomputer output display.

The eighth output of our PROM is used to detect an ASCII space and break it out of the code, for a Selectric space is a machine command, and an ASCII space is a printing character. If a space is detected, the keypressed output is diverted to the space solenoid and the ball moving solenoids are disabled.

If a lower case alphabet is provided, the input logic on bits a6 and a7 detects lower case and converts it to its equivalent upper case ASCII six bit input code and at the same time forces the shift output line to the lower case low state. This is a good example of how a quarter's worth of gating can cut the size of a ROM in half.

Another Selectric machine command we need is a *return* command to move the ball back to the beginning of a line. This command can be detected with the lower left NOR gate (Fig. 6) and used to divert the keypressed output to the carriage return solenoid, at the same time disabling the ball-moving solenoids. If there should be any other control commands which are to be ignored by the typewriter, carriage return detection logic (OPTIONAL CR GATING in Fig. 6) can be added to distinguish carriage returns from the added control commands.

Fig. 8. Full function ASCII to SELECTRIC code converter uses larger PROM.



You can build this ASCII to Selectric interface using a single 64 x 8 PROM or a pair of 32 x 8 PROMs with parallel outputs selected by the chip enable inputs. Using two ICs is sometimes less expensive than one larger PROM because the smaller PROMs are more widely available as surplus. Fig. 5 lists several typical PROM parts.

All Selectric functions including bell, tab, backspace and index are accommodated in the circuit of Fig. 8. It takes a PROM of four times the size, provides more functions, and is a simpler circuit than the one shown in Fig. 6. The eighth PROM output is used to decide whether the ASCII code is a printed character or a machine command (which might be ignored). This eighth output line *feeds back to the PROM's eighth input line* via the resistor-capacitor time delay circuit. When bits 1 to 7 of the PROM input represent a printable character, the eighth bit output line switches to a high level which enables printing.

The ball output codes are the same as in the simpler circuit, but their solenoids must be disabled when the machine command codes are received. When a character is to be printed, the keypressed pulse (KP in Fig. 8) is routed by the upper AND gate (enabled by the logical one output of the PROM's eighth bit) to the print solenoids; the ball solenoid enable is also taken from the eighth PROM bit. The result is movement of the ball combined with a print stroke.

If, however, a machine command is presented to the PROM at the ASCII inputs, the eighth bit of the PROM output goes low and disables both the ball solenoids and the gate which would route KP to the print solenoid. This bit is inverted by the inverter to present a logical one level to the lower AND gate in the figure. This enables the keypressed pulse (KP) to go to the machine solenoids. Any legitimate command code will result in selection of *one* of the machine outputs in Fig. 8. The selected machine output line will enable the corresponding machine control solenoid's driver — resulting in one of the machine control actions such as a tab. As an example, suppose a horizontal tab function (HT in ASCII) is presented to the PROM: The PROM decodes a machine command, inhibits ball solenoids and the print solenoid, presents a decoded logical one level to the tab solenoid driver, enabling it, and routes the keypressed signal (KP) to the other tab solenoid driver input — resulting in a tab action. Since the PROM is to be set up for only six (or seven) legitimate machine commands, any unwanted ASCII machine control codes will be ignored.

In this article, we've seen some background information on ROM technology and several of the many uses to which ROMs can be put in microcomputer and logic systems. These are by no means the *only uses* of ROMs. The uses of ROM technology are for the most part limited only by your own imagination. ■



THE H-P 65: WORLD'S SMALLEST COMPUTER SYSTEM

by
Richard Nelson
HP-65 Users Club
2541 W. Camden Pl.
Santa Ana CA 92704

On Jan. 17, 1974, Hewlett-Packard announced their fourth model in a continuing series of sophisticated pocket calculators. This model, the HP-65, was unique in that it was programmable and used magnetic cards to read and write user-written programs. One of the first questions asked about the new machine was, "Is it a calculator or a computer?" There is no question that the HP-65 is the most powerful computational tool in existence for its size and weight, but in terms of an Altair 8800, Mark I or Sphere microcomputer, how does the HP-65 stack up?

Computer vs. Calculator

In the classical definition of a computer, as described by Babbage, the HP-65 is indeed a computer. The HP-65 has conditional and unconditional branching, data storage, and a memory which can be used to store a program. By today's standards a machine is often called a computer or a calculator by the task for which it is best suited. A calculator is best suited for numerical calculations, and a computer is best suited for

binary or alphabetic data manipulations involving a large data base. In this sense, the HP-65 is – as it is called – a calculator.

If you are interested in computers and programming, should you consider the HP-65? Most definitely! The only negative aspect of the HP-65, compared to a typical microcomputer kit, is its cost (i.e., $795 versus about $500). If, however, you consider the microcomputer complete with an operating system and an input/output device – but built into a pocket-size package, then the HP-65 is a very competitively priced unit.

There is no doubt that the HP-65 "computer" is the easiest to get up and running. If your interests are primarily of a problem-solving nature, and a personal, portable computational and educational tool will meet your needs, the HP-65 is for you.

If your interests tend toward programming, the HP-65 is an excellent machine to learn programming upon. If you master memory limited (every computer is memory limited sooner or later) programming on the HP-65, you will have an excellent background to move up to a larger machine if you desire. There will be no problem selling a used HP-65. Just try buying one!

How Powerful is the HP-65?

The HP-65 has 100 memory steps, and 14 data registers (four in the stack) directly accessible to the user. Programs may be linked from card to card if efficient programming won't allow placing all steps on one card. Card read time is about the same as the dial "0" time on an older model telephone. Four logical comparison operations, two flags, decrement counter, convenient editing, five user-defined keys and merged codes which allow two keystrokes to occupy only one step of memory all combine to make the HP-65 so powerful that it was used to backup the on-board computer on later Apollo missions. Programs have been written for the HP-65 to perform some amazing calculations – and other tasks as well.

Hewlett-Packard has nearly 4,000 programs in their library, all available to HP-65 users. A catalog details and classifies each program. Hewlett-Packard also offers collections of programs in the form of PACs containing 40 pre-recorded cards. To date, 14 PACs are available, covering the fields of finance, mathematics, electrical, chemical and mechanical engineering, medicine, surveying, aviation and navigation.

The HP-65 can be programmed to:

a) Play Tic-tac-toe, NIM, Bagels, Craps, Ping-pong, Slot Machine, Football or Hexapawn. Some games are cybernetic – one card – and will actually learn to play a perfect game.

b) Generate 10,000 digits to test a random number routine, sort and tally the digits 0-9 to check the routine's uniformity.

c) Play word games such as Word Squares or Hangman.

d) Generate a table of prime numbers.

e) Solve up to seven simultaneous equations in seven unknowns.

f) Design a transistor amplifier circuit with all calculated values converted to EIA standard values.

g) Teach students arithmetic using teaching programs that adjust to the learning rate of the student.

h) Compute double precision products such as:

9, 753, 124, 680
x 9, 375, 168, 024 =
91,437,182,634,
021,232,320

(64-bit arithmetic is not enough precision for this number-crunching application.) The latter is an example of the type of problems in which the calculator is outstanding and the microcomputer is marginal. Try an alphabetic sort of the names of the members of your car pool and the situation is reversed.

Like all electronic "computers," someone always applies their machine in unusual ways in an attempt to solve a problem. For blind users a program has been written for the HP-65 which will produce a tone on a radio in the form of coded beeps to "spell out" the display. One user has applied a program to generate points to plot a modern art figure. In the search of trying to get 200 steps into a 100-step memory, users have even found logic that wasn't intended to be included in the instruction set. Many HP-65 users get the most from their machine by sharing their experiences through an organization established for that purpose. Most computers have users groups – for the HP-65 it is the HP-65 Users Club. ■



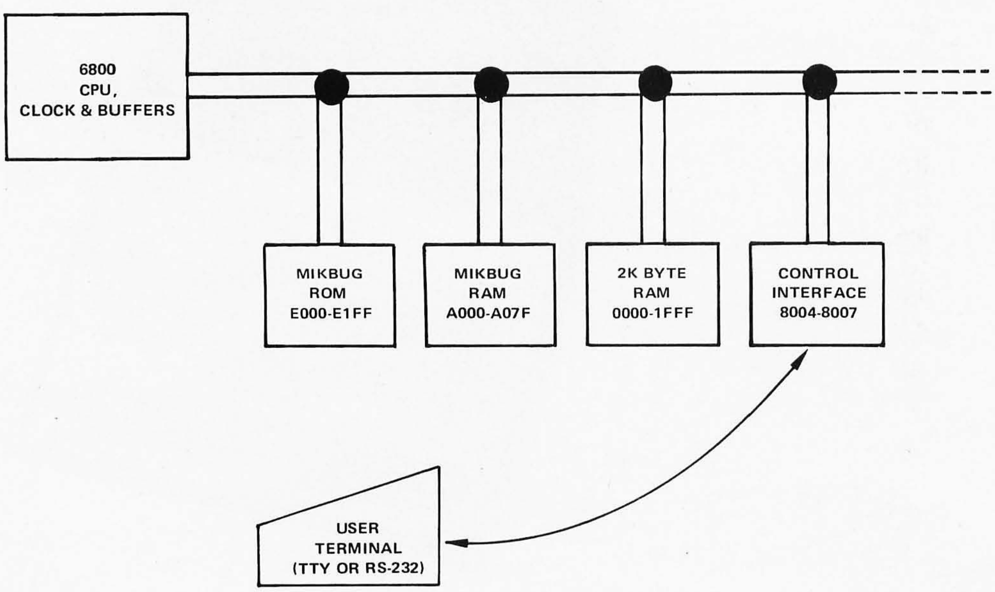
Build A 6800 System With This Kit

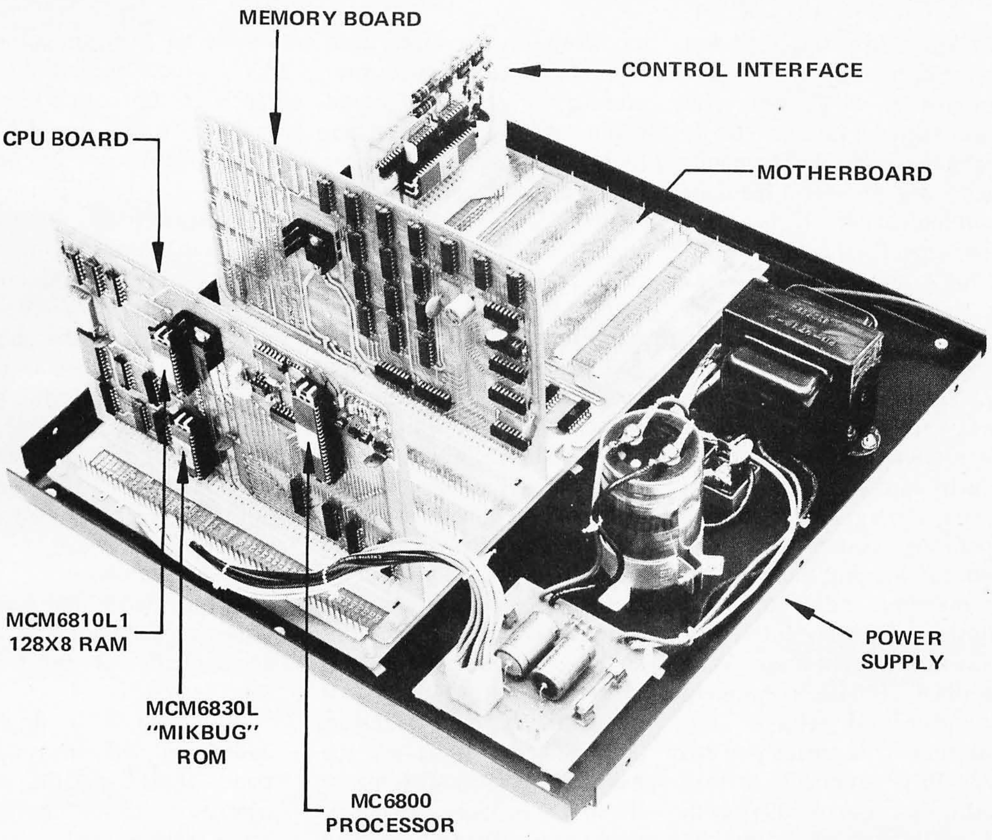
by
Gary Kay
Southwest Technical Products Corp.
219 W. Rhapsody
San Antonio TX 78216

If you are one of the many people getting ready to purchase one of the reasonably priced microprocessor system kits on the market today, you might ask yourself whether or not you will be able to start entering programs once you get it all put together. Of course you can always load programs and data through the front panel programmer's console, but most individuals aware of the front panel's slow speed and difficult readability prefer to use a Teletype or low cost video terminal such as the TV Typewriter II (February 1975, *Radio Electronics*) for data and program input/output. This is all well and good except that in order to attach a terminal, you'll have to purchase an interface for your computer if it is not supplied with the basic system. In fact you will generally need a separate interface for each I/O (input/output) device connected to your computer. This can run your system investment up considerably since such interfaces typically cost between $75 and $150 each, and there are more surprises yet to come.

So now you've got your computer, with interface, attached to your terminal; you're ready to sit down, power up and start typing in your program, right? Well, not quite. You see, in order to be able to use the terminal for either entering programs or getting data in and out of the computer you must have a program resident or loaded into memory telling the processor how and what to do. Without this software (program), you can pound on the keyboard all you want and the computer won't do anything. Computers are no smarter than their programming lets them be and without programming they're not very smart at all. How do you get this software into memory? Well, you could load it in from paper or cassette tape, that is if you have a paper tape reader or cassette tape interface (another sizable investment) or you could enter it directly from the programmer's console. The problem here is two fold. Software to give the terminal reasonable system control will probably be around 500 words in length. This is far too long to enter from the programmer's console especially when you consider it has to be re-entered every time the system is powered up or after a wayward program overwrites any of its allocated area of memory. The second problem is that few if any of the manufacturers supply a listing, paper tape or cassette tape of such a program to begin with. Their terminal control routines are contained within editor/assembler and higher

Fig. 1. Block diagram of the SWTPC 6800 system. The address allocations of the elements of the system are noted inside the blocks.





Details of the SWTPC 6800 System. This photo illustrates what you see when you remove the cover of a typical SWTPC computer system. This is an assembly of the parts which come in the MP-68 kit.

level language packages which not only must be loaded from some kind of tape reader, but require from 4,096 to 8,192 words of memory to operate. And you thought the interfaces were expensive, just check the prices on 8,192 words of memory. Many of the systems now on the market are supplied with an amount of memory with the basic unit which is considerably less than what might actually be needed for useful programming.

So what's the alternative? Well, the system presented in this article has been designed to eliminate the aforementioned problems and allow the user to have a powerful and fully functional system at a minimum cost (see Fig. 1). The entire system is built around the Motorola MC6800 microprocessor and its family of support devices. The computer itself is being made available in kit form including the chassis, cover, power supply and all circuit boards, parts and hardware necessary to build a Motorola 6800 based microprocessor including a 1,024 word ROM (read only memory) stored operating system with 128-word scratch pad memory, serial interface baud rate generator, serial interface, and 2,048 words of memory for $450. This article gives a description of the microprocessor and mother board. A future article will describe the power supply, memory and interface boards.

The Microprocessor/System Board (MP-A)

The Microprocessor/System Board (coded MP-A) is the primary logic board for the system. It is a 5 1/2" x 9" double sided plated-through hole circuit board containing the 6800 microprocessor chip, the 6830 ROM which stores the mini-operating system and the 6810, 128-word scratch pad random access memory (RAM) needed by the ROM. There is a crystal controlled processor clock driver and baud rate generator providing serial interface baud rates of 110, 150, 300, 600 and 1200 baud for all but the terminal control interface which is operable at 110 or 300 baud. Also provided is a power up/manual reset circuit which restarts the ROM stored mini-operating system whenever activated. Full I/O buffering is provided for the 16 address lines and eight bidirectional data lines with these and other connections made to the rest of the system through the mother board via a 50-pin connector. Power for the board is derived from a +5 volt regulator fed from the system's unregulated 7 volt, 10 Amp power supply. Average current consumption for the board is 0.8 Amps.

The mini-operating system stored in the 6830 ROM on this board has got to be one of the most outstanding features of this system. It is through this Motorola written software package called "MIKBUG" that the user can 1) enter program or data into memory from either the terminal's keyboard or tape (where applicable), 2) jump to and execute a program loaded in memory, 3) list programs or data stored in memory, on the terminal or tape (where applicable), 4) examine and/or change the contents of the internal CPU registers, 5) examine and/or change the contents of specified memory locations. These operations are performed using a 20 mA current loop Teletype or an RS-232C compatible serial ASCII terminal.

This ROM mini-operating system does not have to be loaded from tape and it cannot be overwritten. It is always there at your fingertips — just pressing the RESET button or simply powering the system up automatically restarts this firmware (ROM stored software). When activated, this system responds with a

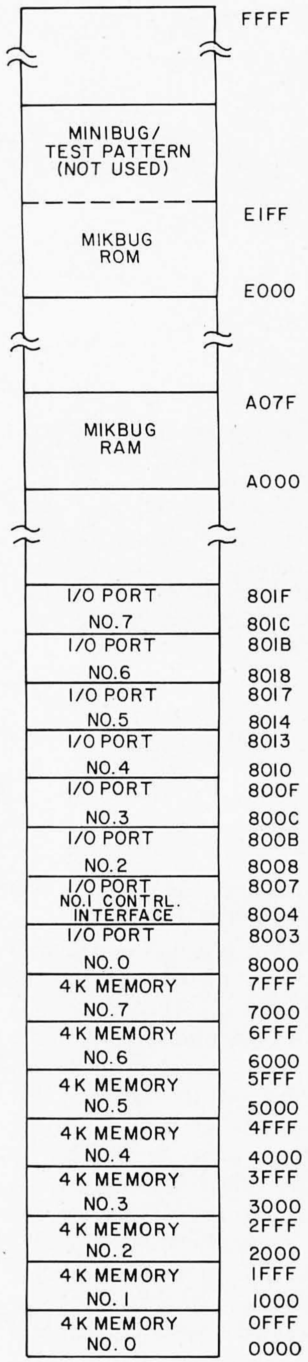


Fig. 2. SWTPC 6800 Microprocessor System memory map. The 64K address space of a 6800 CPU is divided up into the segments shown here. The first 32K locations are available for user read-write memory. The second 32K is devoted to I/O port assignments and the requirements of the MIKBUG program supplied by Motorola.

carriage return, line feed and then prints a * on the terminal at which time you may enter various single character control commands such as M for memory examine/change, L for load from tape, P for punch or list, R for examine registers or G for go to and execute a loaded program. A program debug routine can also be implemented by using the software interrupt (SWI) instruction as a "breakpoint" which forces a jump from your program to the operating system to allow you to examine the contents of memory and/or the CPU registers. All data entered or displayed through the terminal is in convenient hexadecimal (base 16) notation. This means you can type in a command to load address location $A000_{16}$ with $9E_{16}$ instead of setting 24 console switches to an address of 1010 0000 0000 0000 with data of 1001 1110 as must be done with the conventional programmer's console. Since the operating system is stored in ROM, it consumes no user RAM memory, in fact, it actually gives the user a little extra. There is a 128-word scratch pad memory utilized by the operating system for storing various addresses and data, but there are more than 54 locations within this 6810 RAM memory which are totally unassigned plus a 46-word deep push-down stack. All of this memory is in addition to the 2,048 words (expandable to 4,096 words) contained on the standard memory board.

Since the terminal and mini-operating system provide the user with complete system control, there is no need for the conventional programmer's console. Take note also that once system control is turned over to your program, the control terminal is totally available for your program input/output. In fact, since the character input/output subroutines are already stored within the operating system ROM, they can be used by your programs simply by loading or storing the characters to be handled in the proper register and executing a jump to subroutine (JSR).

The Motorola MC6800 microprocessor chip is the element around which this entire system is built. It is an 8-bit parallel processor with eight bidirectional data lines and 16 address lines giving it an addressing capability of up to 65,536 words. There is no distinction between memory and I/O addressing on this system, therefore, all input/output data transfers are handled just as are the memory transfers. This means the I/O interfaces must have their own allocated memory addresses where neither ROM or RAM memory may be located. This may at first seem to be a disadvantage until you realize that all memory handling instructions are usable for the interface data handling as well, thus eliminating the need for special data I/O instructions. The memory assignments for this system have to be made as shown in Fig. 2. User RAM may be located anywhere in the lower 32K (0000_{16} to 8000_{16}) addresses with the upper 32K addresses reserved for the operating system ROM, RAM and interface boards.

There are six registers internal to the MC6800 microprocessor element which consist of the program counter, stack pointer, index register, accumulator A, accumulator B and condition code register. The stack pointer is a 16-bit register used to store the address of the push-down stack which is located in RAM memory external to the MC6800 microprocessor element. The push-down stack itself is used to store the program counter and/or processor data during branch to subroutine (BSR), jump to subroutine (JSR), push (PHS) or interrupt routines. The index register is a 16-bit register generally used as an address pointer for many processor instructions.

There are 72 basic instructions for the 6800 microprocessor system (Fig. 3) with most of the 72 utilizing several of the seven possible addressing modes: Accumulator, implied, relative, direct, immediate, extended and indexed.

- *Accumulator* — In accumulator addressing, either accumulator A or accumulator B must be specified.
- *Implied* — In implied addressing the instruction code itself specifies the operand (stack pointer, index register, etc.).
- *Relative* — Relative addressing is used for the branch instructions and indicates the value contained in the word of memory immediately following the instruction code added to the program counter +2 with the result then loaded back into the program counter. Positive data (bit 7 = 0) generates forward jumps up to 129 words from the branch instruction while negative data (bit 7 = 1) generates backward jumps up to 125 words from the branch instruction.
- *Direct* — In direct addressing, the value contained in the word of memory immediately following instruction code is an actual memory address within the first 256 words of memory (0000_{16} to 00FF) which contains the operand of the instruction. This mode typically saves one CPU cycle of execution when compared to *extended* addressing.
- *Immediate* — In immediate addressing, the

value contained in the word, or in some cases two words of memory, immediately following the instruction code is the operand of the instruction.

- *Extended* – In extended addressing, the two words of memory immediately following the instruction code contain the address of the memory location which contains the operand of the instruction.
- *Indexed* – In indexed addressing, the value contained in the word of memory, immediately following the instruction code, is temporarily added to the contents of the index register generating a new address where the operand of the instruction is located. The jump is positive only, going from 0 to 255 words and the actual contents of the index register are not changed.

Also provided on the main processor board is an MC14411 baud rate generator which uses an external 1.8432 MHz crystal and internal oscillator and divide chain to generate serial interface clocks for baud rates of 110, 150, 300, 600 and 1200 baud. Also derived from this circuit is the 921.6 kHz clock used by the MC6800 microprocessor element. It is first, however, fed into a gating circuit generating two non-overlapping, 50% duty cycle, complementary clock signals ϕ_1 and ϕ_2.

Mother Board (MP-B)

The Mother Board (coded MP-B) is a 9" x 14" double sided, plated-through hole circuit board onto which all of the various processor boards are plugged. Provisions have been made for one Microprocessor/System Board, up to four 4,096 word random access memory boards plus two unused slots. This allows the system to be expanded to 16,384 words of memory. For those demanding even more memory, the 50-line system information bus may be paralleled onto another mother board with separate power supply expanding the system to a maximum of 32,768 words of random access memory.

The Mother Board also provides the line buffering and address decoding for up to eight interface boards. Although one of the eight must be the serial terminal, control interface, the other seven may be any combination of parallel or serial interfaces the user may choose to have. For those demanding even more interfacing capability, the 50-line system information bus may be paralleled onto another mother board with separate power supply expanding the interfacing capability to one terminal, control interface plus any combination of up to 15 serial or parallel interfaces.

The following is a brief description of each of the 50 lines on the system information bus:

The A0 – A15 lines carry address bits 0 through 15 respectively, forming a 16-bit address which is used to define either a memory location or interface address.

The BUS AVAILABLE line goes high acknowledging a processor halt, meaning the processor has stopped and that the system information bus is available for external control.

The $\overline{D0}$ – $\overline{D7}$ lines carry inverted data bits 0 through 7 respectively, forming 8-bit data words which are exchanged between the various boards within the system.

The GND line is the system's common power supply ground point.

Fig. 3. The 6800 microprocessor's instruction set. This is a list of the mnemonics available. A more complete explanation of the basic operations of the processor is found in Motorola's programming manual for the 6800 which is part of the SWTPC documentation package.

ABA	Add Accumulators
ADC	Add with Carry
ADD	Add
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
BCC	Branch if Carry Clear
BCS	Branch if Carry Set
BEQ	Branch if Equal to Zero
BGE	Branch if Greater or Equal Zero
BGT	Branch if Greater than Zero
BHI	Branch if Higher
BIT	Bit Test
BLE	Branch if Less or Equal
BLS	Branch if Lower or Same
BLT	Branch if less than Zero
BMI	Branch if Minus
BNE	Branch if Not Equal to Zero
BPL	Branch if Plus
BRA	Branch Always
BSR	Branch to Subroutine
BVC	Branch if Overflow Clear
BVS	Branch if Overflow Set
CBA	Compare Accumulators
CLC	Clear Carry
CLI	Clear Interrupt Mask
CLR	Clear
CLV	Clear Overflow
CMP	Compare
COM	Complement
CPX	Compare Index Register
DAA	Decimal Adjust
DEC	Decrement
DES	Decrement Stack Pointer
DEX	Decrement Index Register
EOR	Exclusive OR
INC	Increment
INS	Increment Stack Pointer
INX	Increment Index Register
JMP	Jump
JSR	Jump to Subroutine
LDA	Load Accumulator
LDS	Load Stack Pointer
LDX	Load Index Register
LSR	Logical Shift Right
NEG	Negate
NOP	No Operation
ORA	Inclusive OR Accumulator
PSH	Push Data
PUL	Pull Data
ROL	Rotate Left
ROR	Rotate Right
RTI	Return from Interrupt
RTS	Return from Subroutine
SBA	Subtract Accumulators
SBC	Subtract with Carry
SEC	Set Carry
SEI	Set Interrupt Mask
SEV	Set Overflow
STA	Store Accumulator
STS	Store Stack Register
STX	Store Index Register
SUB	Subtract
SWI	Software Interrupt
TAB	Transfer Accumulators
TAP	Transfer Accumulators to Condition Code Reg.
TBA	Transfer Accumulators
TPA	Transfer Condition Code Reg. to Accumulator
TST	Test
TSX	Transfer Stack Pointer to Index Register
TXS	Transfer Index Register to Stack Pointer
WAI	Wait for Interrupt

The normally high $\overline{HALT}$ line when brought low halts the processor and frees the system information bus for external control.

The INDEX line is an unused one and is provided so the pin on each of the male connectors may be cut with the corresponding female connector pins plugged, preventing the circuit boards from being plugged on incorrrectly.

The $\overline{IRQ}$ is the maskable, single level interrupt request line feeding the processor board. If not inhibited by software it will when momentarily given a TTL zero level signal, force the processor into a push-down stack store routine followed by a program jump to a user selected address stored in the operating system RAM.

The M. RESET line, when momentarily grounded manually, indirectly resets the registers internal to the processor and interfaces, and loads the ROM stored mini-operating system. This line is normally grounded by depressing the RESET button on the system's front panel.

The $\overline{NMI}$ is the non-maskable, single level interrupt line feeding the processor board. When momentarily given a TTL zero level it forces the processor into a push-down stack store routine, followed by a program jump to a user selected address stored in the operating system RAM. The $\overline{NMI}$ is not maskable thus cannot be inhibited by the programmer through software.

$\overline{\phi 2}$ is one of the two complementary system clock outputs and is used to signal that valid data is on the data lines $\overline{D0}$ – $\overline{D7}$ when low.

$\phi 1$ is the non-overlapping clock complement of $\phi 2$.

The $\overline{RESET}$ line when low resets the registers internal to the processor and interfaces, and loads the ROM stored mini-operating system. This line is activated by one shot on the Microprocessor/System board when the system is first powered up or when M. RESET line is momentarily grounded.

The R/$\overline{W}$ line establishes the direction of data flow on the eight data lines, $\overline{D0}$ – $\overline{D7}$. It is high for a read from memory or interface and is low for a write to memory or interface.

$\overline{VMA}$ is the valid memory address line which goes low to confirm that valid memory address data is being presented on the 16 address lines, A0 – A15.

The UD1 and UD2 are user defined lines and have not been assigned a function.

The –12 and +12 points are lines to which an isolated ground –12 @ 200 mA and +12 @ 200 mA power supply should be connected. The voltages are necessary for generating the currents required by 20 mA current loop Teletype equipment on the serial interfaces.

The 7 – 8 VDC UNREG point is the line to which a +7 to 8 volt dc @ 10 Amp unregulated power supply should be attached. This voltage is then regulated down to +5 V dc by independent regulators on the various boards within the system.

The five 110b, 150b, 300b, 600b, 1200b lines carry 1758.8, 2400, 4800, 9600 and 19200 Hz clocks required by the serial interfaces for 110, 150, 300, 600 and 1200 baud communication.

Also attached to the 50-line system information bus are the interface decode and driver circuits. A considerable cost savings is made here by providing the address decoding and information bus buffering for all of the interfaces right on the mother board instead of providing it on each of the interface boards individually. Since each of the parallel interfaces require four address locations and the serial two, four addresses are provided for each of the interface positions. They are assigned as shown in the memory map, Fig. 2. Interface position 1 (8004 – 8007) is reserved for the terminal control interface. The signals carried on the interface information bus are almost identical to those on the system bus. UD3 and UD4 are here again User Defined data lines and RS0 and RS1 are Register select lines which are identical to address lines A0 and A1 respectively. Power for the address decode and buffer circuits on the mother board is provided by a separate on board regulator with a current consumption of typically 0.4 Amp. ■

(More SWTPC 6800 data is coming in BYTE.)



Once you've assembled and checked out the operation of your MP-68 kit, the result will be a product which looks like this. Note the complete absence of most of the usual control panel functions you might expect. This is achieved by using a serial communications device such as a Teletype or an RS-232C compatible terminal as the "front panel."

COMPUTER EXPERIMENTER SUPPLIES

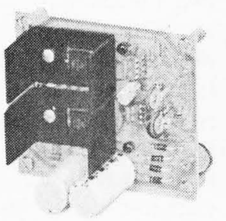
FACTORY FRESH—PRIME QUALITY
PERFORMANCE GUARANTEED

MICROPROCESSORS AND MEMORY

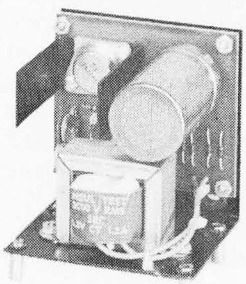
8008	$ 35.00	Commercial Grade—up to 35°C.
8080	135.00	These units are factory fresh, full spec devices.
2102	3.50	These units are factory fresh, full spec devices.
2102-2	4.50	These units are factory fresh, full spec devices.

COMPUTER GRADE REGULATED POWER SUPPLIES

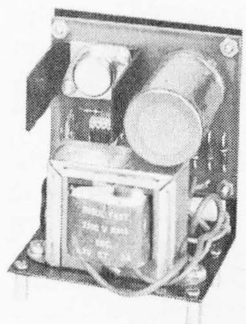
All units are short-circuit proof, fold back current limited and with over-voltage crowbar protection.



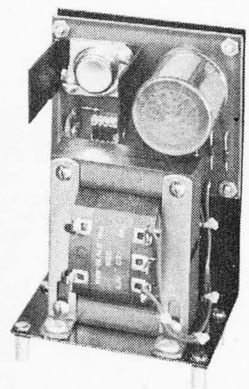
MD-15
±15 Volt at 200MA
Dual Tracking
$30.00



MD-5-1
+5 Volt at 1 Amp
$24.50



MD-5-3
+5 Volt at 3 Amp
$34.50



MD-5-6
+5 Volt at 6 Amp
$44.50

MICRO COMPUTER SUPPLY COMBINATIONS

For the 8008
MD-08—+5 volt at 6 amp, -12, -9 at 200 ma $75.00
For the 8080
MD-80—+5 volt at 6 amp, ±12v at 200 ma ... $75.00
For the Fairchild F-8
MD-8—+5 volt at 6 amp, +12 v at 200 ma ... $65.00
For the M6800
MD-5—+5 volt at 6 amp $44.50

All units are short circuit proof, fold-back current limited and with over voltage crowbar protection.

TTL INTEGRATED CIRCUITS

All devices are factory fresh, full spec units.

7400	.23
7404	.25
7442	.60
7447	.95
7448	.95
7475	.60
7490	.60
7493	.60
74125	.55
74126	.55
74192	1.10
74193	1.10

GUARANTEE
Most devices shipped within 24 hours. If not shippable within 2 weeks payment refunded. Performance guaranteed on all units for 30 days. Defective parts replaced at no charge. NOTICE: This warranty applies **only** to parts that have not been soldered. You must use sockets for your incoming inspection tests.

All Prices Subject to Change Without Notice
Minimum Order $10.00
Add $1.00 to Cover Postage and Handling
Send Check or Money Order (No C.O.D.) To:
N. J. Residents Add 5% Sales Tax

MICRO DIGITAL CORP.

BOX 413, EDISON, NJ 08817 • (201) 549-2699

First Person Report:

Assembling an Altair 8800

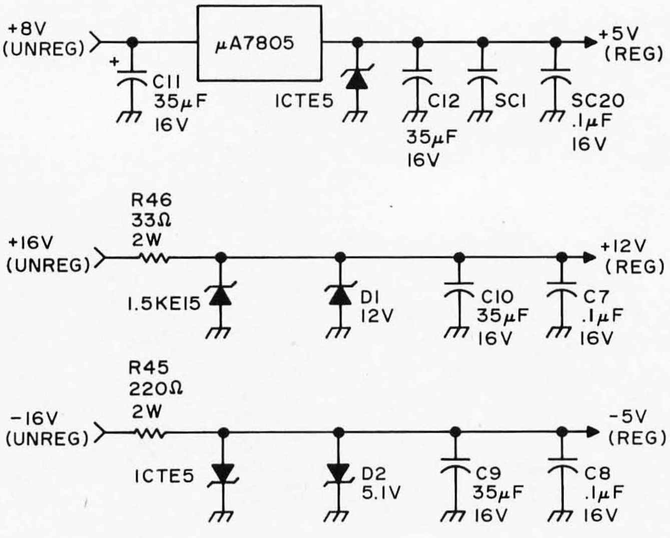
by
John Zarrella
90-9 Wakelee Rd.
Waterbury CT 06705

My adventure with microprocessors began rather late in the hobby game, at the end of 1974. It was about this time, or so it seemed to me, that micros became the topic of conversation in anything related to computers and automation. With the IMP-16, the 8080, 8008, 4004, etc., it became clear that this was what the computer market was waiting for. However, it was the article on the MITS Altair in the January 1975 issue of *Popular Electronics* which finally did it. Although inaccurate and vague, it certainly decided me — I was definitely going to own a micro. The next few months saw hurried mailings of information requests to any company which produced a product even remotely connected with a microprocessor. I immediately got out my checkbook, and mailed all my hard earned dollars to every newsletter that was published, in my frantic search for the "right" processor.

The results were both rewarding and disappointing. I found that there were some fantastic processors, but since my hardware background is not all that hot, I decided that I would have to opt for a kit with one of the most powerful micros I could find. I figured that this would enable me to get on line quickly, learn enough hardware to keep up with the state-of-the-art, and permit me to evaluate new micros as they came out, so I could build my "dream machine" when the right parts became available.

I decided to build the Altair 8800. Although the instruction set looked rather impressive, what convinced me was seeing a process control system which used the 8080; I was truly impressed with its capability.

Fig. 1. The schematic diagram of power supply circuitry, showing additional protection diodes.



I decided I would have to opt for a kit . . . this would enable me to get on line quickly.

The Order

After calling in my order to MITS, I waited nearly seven weeks for delivery. MITS did make it within the advertised 60-day delivery time. All was not roses for those seven weeks, however; it seems that either MITS or BankAmericard got their signals crossed and couldn't get a credit authorization (they both eventually declined to accept responsibility). You can imagine what it was like getting a call during dinner, explaining that my unit was

ready to ship, but unfortunately . . . Luckily they agreed to ship it COD, and I quickly ran down to my bank to get a certified check. Every morning I left my wife with the admonition not to miss the delivery, and every day at lunch I called to determine whether or not my "computer" had arrived. (Did you ever try to ask your insurance agent whether you needed extra renter's insurance – "You keep a computer at home??!! What for?")

Assembly

Within a week of that call, I had the Altair in my hot little hands. "Are those little plastic parts all you get for $500.00?", my wife exclaimed, peering over my shoulder. Undaunted, I shooed her out and locked myself in the back room all weekend soldering PC boards. It took three weekends to complete the assembly (was it my fault I came down with pneumonia in the middle?).

Ah yes, assembly. In general, I found that the MITS assembly instructions were well written. However, their additions were sometimes in the manuals in the wrong place (e.g., page 68A after 69). In at least one case (front panel control board) I had already tightened the panel in place (bolts on numerous switches), when I read that the nut on the little screw holding the voltage regulator to the board (accessible only with the panel out) had to be removed to add a grounding strap. Therefore it pays to check the manual pages carefully, and look two or three pages ahead to see if there are any little tricks sneaking up on you.

As for the parts, only one resistor was missing; however, out of all the screws and bolts supplied with the kit, I could never find the right one to fit. Maybe it was my own stupidity, but it seemed that the last bolt of any given size was always supposed to be used in at least 10 more places. I found that it pays to have a good assortment of screws and bolts (number 6, various lengths ¼" to ¾") to permit frustrationless assembly.

Of all the assembly, the worst (and easiest to mess up) part was correctly connecting the 60 bus wires between the display/control board and the chassis motherboard.

All soldering and component placement was easily accomplished – positions were clearly marked on the boards and in the manual. This is high praise since I hadn't built many kits before; and of these, none were this large. Of all the assembly, the worst (and easiest to mess up) part was correctly connecting the 60 bus wires between the display/control board and the chassis motherboard. I used an Ohmmeter to assure that each connection was correct and that there were no solder bridges to the other bus lines. There's got to be a better way. I hear Processor Technology, Inc., is currently marketing a 16-slot motherboard (on the Altair you have to jumper four of the four slot boards together, only one of which comes with the kit), and an improved connector for the display/control board. These will definitely be my first additions.

I made only one modification to the circuit during assembly. That modification was to add three protection zeners to the CPU board. Fig. 1 shows the electrical connections for this change. These were inserted to protect the 8080 chip (still pretty expensive in singles) from power supply failure. These zeners should ground out overvoltages at currents up to 100 Amps. ICTE5s were used for the +5 V and –5 V lines to the 8080 and a 1.5KE15 for the +12 V. The zeners on the CPU board are illustrated in Fig. 2.

I also added sockets for the 8101 RAMs, cleaned all boards with trichloroethylene solvent, and inspected the finished boards with a magnifying glass. I would highly recommend these procedures as they helped me find more than one solder splash and cold solder joint.

Did you ever try to ask your insurance agent whether you need extra renter's insurance for a computer?

The Big Test

On the fourth weekend I got up the courage to mount the 8080 and 8101s. Then came turning on the power and checking voltages. Everything looked good, with very little ripple from the

Fig. 2. Detail of the additional protective diodes mounted on the Altair CPU board.

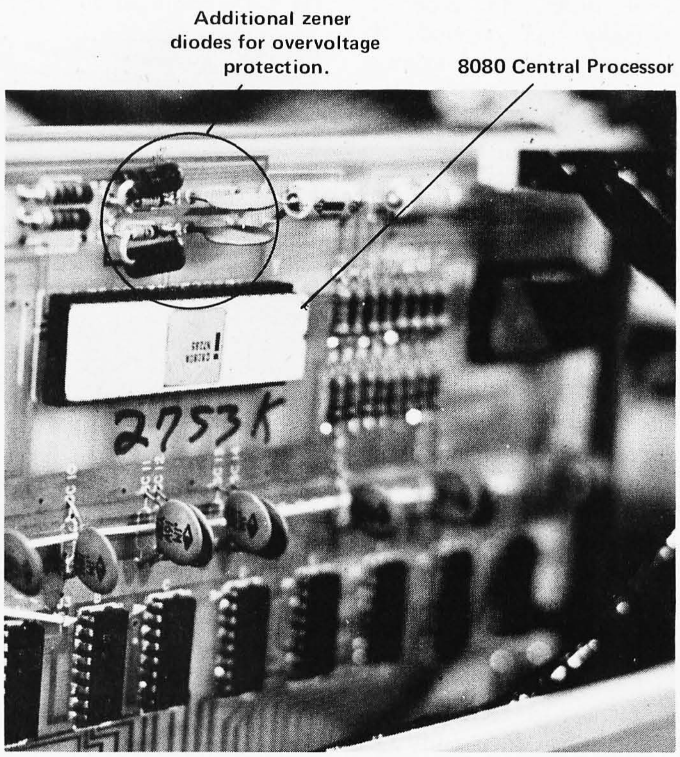
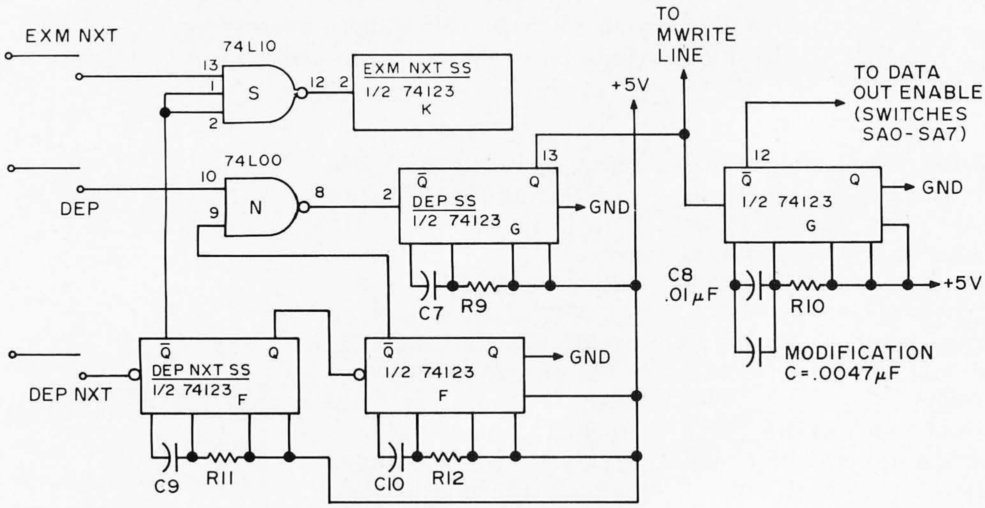


Fig. 3. Adding a parallel capacitance of .0047 uF to C8 of the Altair CPU board schematic lengthens the data out enable line time so that memory write does not extend longer than the data out time.



on-card voltage regulators. Finally the big test: Run a program. This is where the only problem finally showed up. I stopped and reset the CPU, set the switches for my spectacular program (JMP 0) and would you believe it, "deposit" wouldn't work. An hour later I had determined that all other panel switches worked correctly (including deposit next), and that the deposit switch itself was in good order. In order to initially get around the problem I had to examine location 177777 (all address bits 1), then use deposit next to get to location 0.

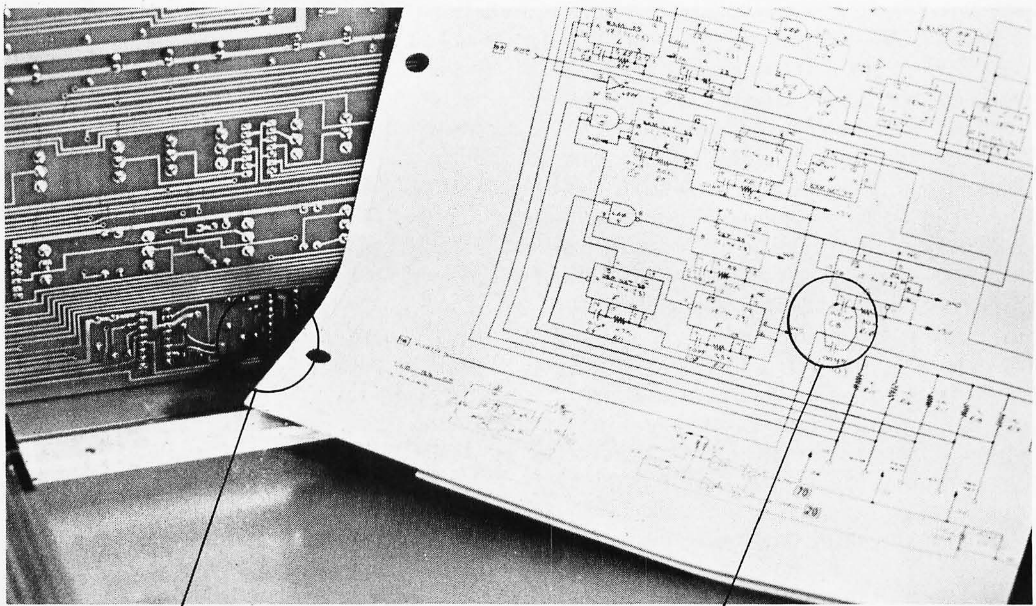
A study of the schematics showed that deposit and deposit next use the same circuitry, except that deposit next first does an examine next. You can verify this visually by loading all ones into the first 10 locations of memory. Then, if you use deposit next to change all the locations to zero, by carefully watching the data LEDs, you will notice that they all flash on as the switch is activated (examine next) and immediately go off again as the deposit is performed.

I concluded that the problem had to be in the timing, since the circuits were otherwise identical. Sure enough, when I looked at the signals on a scope, lo and behold, when a deposit was performed, the memory write line was enabled for approximately 20 ns more than the data out line. There are two oneshots in the deposit circuit; the first enables the memory write line, and the second enables the data out line. The memory write problem was cured by increasing the capacitance on the second deposit oneshot. An increase of .0047 uF (which increases the data out enable time by at least 30 ns) proved sufficient. This was obtained by adding the .0047 uF capacitor as shown in Fig. 3. When building the Altair, this means that C8 (front panel control board) should be approximately .0147 uF; if the board is already assembled into the case, a .0047 uF capacitor can easily be soldered onto the back of the board without removing any components from the case. (Be sure to unplug the computer before making the change, however.) Fig. 4 shows placement of the new capacitor and the change to the Altair schematic diagram.

I feel that the kit is reasonably well made and a good buy – at least at the current 8080 single lot prices, though the add-on options may cost somewhat more than elsewhere.

My plans for my unit currently involve addition of vectored interrupts (a 9318 or 74148 8-bit to 3-bit priority decoder is about all that's needed to translate the eight vectored interrupt lines on the bus into an RST instruction), a real-time clock, monitor clock and some type of I/O (teletype, CRT, etc.). ■

Fig. 4. The additional .0047 uF capacitor is mounted on the rear of the control panel board.

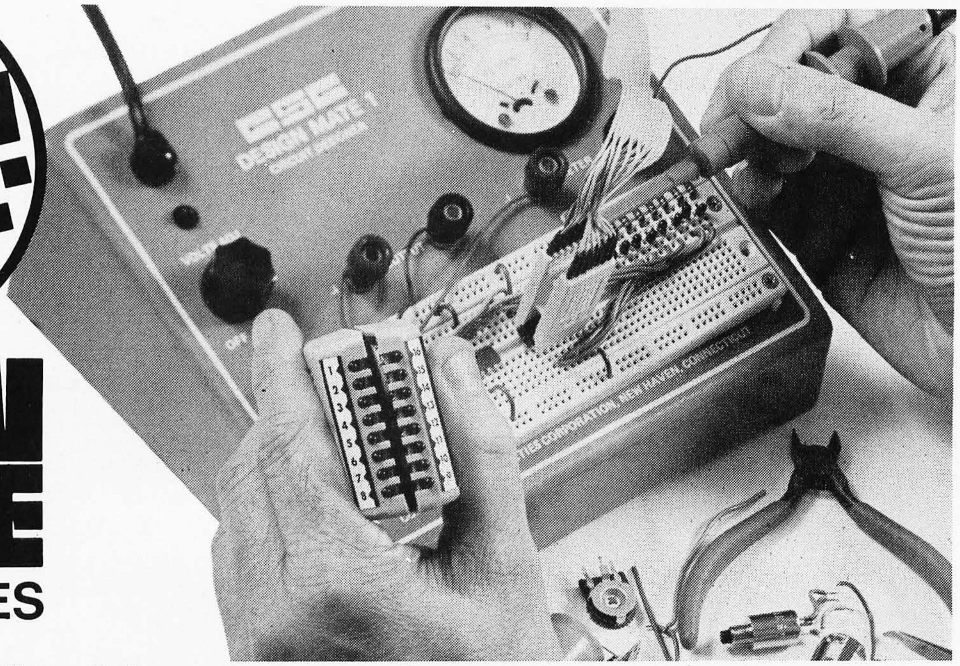


Solder the additional capacitor to the rear of the control panel board.

Modify this section of your schematic.

NEW!

THE
DESIGN MATE
SERIES



At last! High quality, laboratory-grade test instruments . . . for the professional and hobbyist . . . at prices everyone can afford!

DESIGN MATE 1
CIRCUIT DESIGNER

Now you can build/test electronic circuits WITHOUT SOLDER . . . using solid #22 AWG wire to interconnect discrete components . . . resistors, transistors, linear/digital ICs in TO5 or DIP packages (8-40 pins), and more. Plus, you get 5-15VDC up to 600ma (9 watts) of variable regulated power, with a built-in 0-15V voltmeter to monitor internal power and/or external circuits. Now, that's design flexibility! And look at the low, low price!

49.95
Add $2.50 shipping/handling



SPECIFICATIONS
Power Supply: Output: 5-15V @ 600ma. **Ripple and Noise:** less than 20mv @ full load. Load/Line regulation: <1%. Meter: 0-15V DC. **Connectors:** 1 QT-59S, 2 QT-59B, 2 power supply 5-way binding posts, 2 meter 5-way binding posts. **Wght:** 3 lbs. **Power Needed:** 117V, AC @ 60 Hz 12W.*
Patent #235,554.

64.95



Add $2.50 shipping/handling

DESIGN MATE 2
FUNCTION GENERATOR

Troubleshooting? Design Testing? DM-2 gives you all the signal source capacity you need . . . at a very modest price. This 3-wave form Function Generator has: short-proof output, variable signal amplitude and constant output impedance. Completely wired, tested, calibrated, ready to test audio amplifiers, op-amp and educational lab designs . . . as well as complex industrial lab projects. Complete with easy-to-read instructions/operations manual, application notes, operation theory and more, DM-2 works hand-in-hand with DM-1 for total versatility.

SPECIFICATIONS
Frequency Range: 1Hz-100KHz (5 ranges: 1-10Hz, 10-100Hz, 100-1000Hz, 1-10KHz, 10-100KHz). **Dial Accuracy:** Calibrated @ 10Hz, 100Hz, 1KHz, 10KHz, freq. accurate to 5% of dial setting. **Wave Forms:** Sine <2% THD over freq. range. **Triangle** wave linearity, <1% over range. **Square** wave rise/fall <0.5 microseconds — 600Ω-20pf termination. **Output Amplitude:** (all wave forms) variable-0.1V-10V peak to peak into open circuit. **Output Impedance:** 600Ω-constant over ampl./freq. ranges. **Wght.:** 2 lbs. **Power Needed:** 117V, AC @ 60Hz 5W.*

DESIGN MATE 3
R/C BRIDGE

Have you been bugged by color codes or unreadable component markings? Forget it! DM-3, the low cost R/C Bridge, measures true component values . . . in seconds . . . to better than 5%. And, it's all done with only 2 operating controls and a unique solid-state null detector, to zero-in on exact component selection . . . instantly! Completely wired, calibrated and tested, DM-3 includes an extensive instruction/applications manual, and operational theory too.

SPECIFICATIONS
Resistance Range: 10Ω-100 megΩ. (6 Ranges: 10-100Ω, 100-1000Ω, 1K-10KΩ, 100K-1 megΩ, 1 megΩ-10 megΩ) **Capacitance Range:** 10pFd-1mFd- (5 Ranges: 10-100 pFd, 100-1000pFd, .001-.01 mFd, .01mFd-.1mFd, .1- 1.mFd.) **Null Detector:** 2 hi-intensity LEDs-hi/lo markings. **Accuracy:** <5% of null dial, range switch setting. **Wght.** 2 lbs. **Power Needed:** 117V, AC @ 60Hz 3W.*

54.95
Add $2.50 shipping/handling



Each measures 6.75″L x 7.5″W x 3.25″ H.; completely assembled, ready to start testing at once. Order your DESIGN MATES today!
**220V @ 50Hz available at slightly higher cost.*

All DESIGN MATES are made in USA; available off-the-shelf from your local distributor. Direct purchases from CSC can be charged on Bank Americard, Master Charge, American Express. Plus, you get a FREE English/Metric Conversion Slide Rule with each order. Foreign orders please add 10% for shipping/handling. Prices are subject to change.

CSC
CONTINENTAL SPECIALTIES CORPORATION
44 Kendall Street, Box 1942, New Haven, CT 06509 • 203/624-3103
West Coast Office: Box 7809, San Francisco, CA 94119 • 415/421-8872
Canada: Len Finkler Ltd., Ontario

Can YOUR Computer Tell Time?

by
James Hogenson
Box 295
Halstad MN 56548

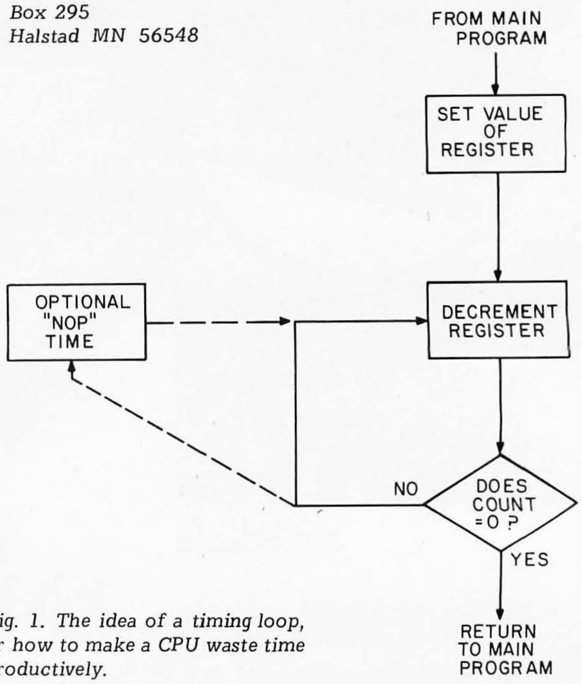


Fig. 1. The idea of a timing loop, or how to make a CPU waste time productively.

Loops are the basic time delay elements. Then there are loops within loops, loops within loops within loops and so on ad infinitum.

Can your computer tell time? O.K. Now take away the LSI clock chip, pocket watch, grandfather clock, or whatever else you managed to interface together. Can your computer still tell time? You bet it can!

It is a readily accepted fact that almost any type of hardware logic device can be imitated or simulated by computer software. That can also include timing devices if you wish.

We will examine a few methods and considerations for software timing, then apply what we've learned in making a novel "software only" clock which will keep time as well as any conventional clock.

The most efficient method (efficient referring to memory space used) to produce a time delay is the use of a loop. This loop is basically very simple, as shown by Fig. 1. By including NOPs or other non-functional time wasters in the loop, the loop can be significantly stretched.

An 8008 is being used in the examples in this article, but the principles hold for any computer. Only the numerical values will change.

The loop represented by Fig. 1 for an 8008 would be a simple three instructions (six bytes) long.

```
LBI  ]  LOAD DELAY
"X"  ]
DCB     DECREMENT "X"
JFZ
L       JUMP BACK
H       UNLESS X = 0
```

The value of "x" loaded into the B register will be the main factor in varying the time delay provided by this loop. Calculating the exact time period is done by tabulation of instruction execution times. These examples will be based on the 8008 instruction execution times with the clock running at exactly 500 kHz.

To calculate the time for this loop, assume the value of "x" to be 1 so no part of the loop is repeated. Add up the number of microseconds required by each instruction.

```
LBI  =  32 US
DCB  =  20 US
JFZ  =  36 US
        88 US
```

Now go back to determine how many microseconds each repetition of the loop will produce. The LBI instruction is not repeated. Do not count any unrepeated instructions in this second tabulation.

```
DCB  =  20 US
JFZ  =  44 US
        64 US
```

Note the different execution times for the JFZ instruction. For the 8008, the execution time of conditional instructions depends upon the condition. If the condition results in a true branch, the instruction takes the longer of the two execution times. The false branch is the shorter time.

The time formula for this loop is

$$64X + 24 = N$$

"x" being the value loaded into B and "n" being the total execution time in microseconds. The unreduced formula is

$$(X - 1)64 + 88 = N$$

Since 64 us are added for each repetition, we must multiply 64 by one less than the value of "x."

255 is the largest possible value of "x" since we are limited to an 8-bit word. Therefore, the maximum time delay that can be provided by this loop is 16344 us. This loop can be stretched by placing a NOP instruction (op code 300) before the DCB, and re-routing the jump.

```
LBI   SET VALUE OF "X"
"X"
NOP   ABSORB EXTRA 20 US
DCB   DECREMENT "X"
JFZ   JUMP BACK TO NOP
L     UNLESS X = 0
H
```

If desired, more than one NOP may be inserted. Each NOP will add another 20x microsedonds. The maximum time with one NOP is 21444 us, the NOP adding 5100 us.

If a timing loop is to be used a number of times at

various points in a program, it may be desirable to rewrite the loop as a called subroutine. The basic flowchart remains unchanged; only the method of implementing it changes.

```
      (MAIN
      PROGRAM)
      CAL    CALL
      L      TIME
      H      LOOP

(TIME
 LOOP)
   LBI       SET VALUE OF "X"
   "X"
   DCB       DECREMENT "X"
   RTZ       RETURN IF "X" = 0
   JMP
   L         JUMP BACK TO DCB
   H
```

Tabulation will show that the basic loop is good for 116 us with each repetition adding 76 us. The reduced formula is

```
76X + 40 = N
```

This loop is a little more complex. Although the CAL instruction which calls the loop is not a part of the loop itself, the execution time of the CAL instruction is a part of the time period produced. We, therefore, must add 44 us for the CAL.

As done before, we assume the value of "x" to be 1 for the first tabulation. The RTZ will be a true branch, so we stop adding there. An RTZ true branch will take 20 us, while an RTZ false branch will take 12 us.

Each repetition will add 12 us for the RTZ, 44 us for the JMP, and 20 us for the DCB instruction. The unreduced formula is

```
(X - 1)76 + 116 = N
```

NOPs placed before the DCB instruction will have the same effect as in the first loop, an additional 20x us per NOP.

The maximum time period produced by this second loop with one NOP is 24520 us. The minimum time period without any NOPs is 116 us. Anything under 116 us can be more efficiently implemented with straight NOPs than with a loop, should such a need arise.

If a time period much longer than 24000 us is needed, modify the time loop to make a double loop as shown in Fig. 2. Make an identical loop, but rather than using a NOP for more time, insert an entire loop.

```
   CAL   (TIME LOOP)
   L
   H

(TIME LOOP)
   LCI        SET VALUE OF "Y"
   "Y"
 A NOP
   LBI        SET VALUE OF "X"
   "X"
 B NOP
   DCB        DECREMENT "X"
   JFZ
   L          JUMP BACK TO DCB
   H          IF "X" ≠ 0
   DCC        DECREMENT "Y"
   RTZ        RETURN IF "Y" = 0
   JMP
   L          JUMP TO LBI
   H
```

Time calculations for multiple loops become somewhat more complex, but again the same principle is used.

The inside loop used here is the same loop first calculated at the beginning of this article. When calculating the main loop, the inside loop is treated as one combined unit of value. The tabulation will look like this:

```
MAIN LOOP:
   CAL   =   44 US
   LCI   =   32 US
   INSIDE
   LOOP  =  (64X + 24) US
   DCC   =   20 US
   RTZ   =   20 US
            (64X + 140) US

EACH REPETITION OR
TRUE BRANCH WILL ADD:

RTZ   =   12 US
JMP   =   44 US
INSIDE
LOOP  =  (64X + 24) US
DCC   =   20 US
         (64X + 100) US
```

The formula, unreduced, would be

```
64X + 140 +
(Y-1)(64X + 100) = N
```

Reducing the formula gives us

```
64XY + 100X + 40 = N
```

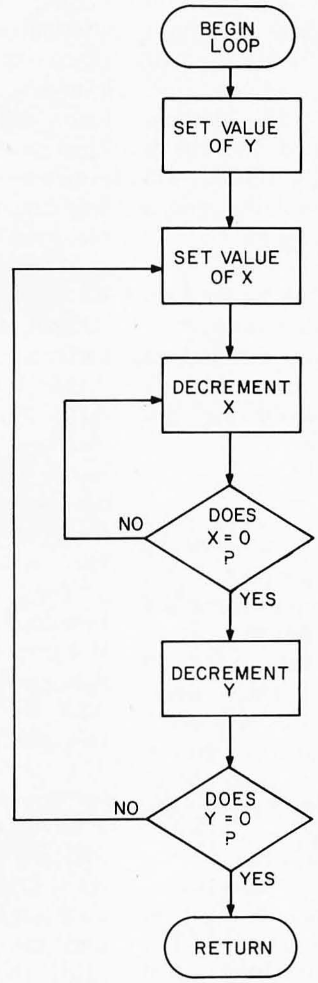


Fig. 2. Getting fancy. By nesting one timing loop within an outer loop, much longer delays can be obtained. Two parameters "x" and "y" are required to completely specify this loop. In a 16-bit machine, of course, the same result (here intended for an 8-bit 8008) can be obtained without nested loops since the 16-bitter can count much higher.

The maximum time delay provided by this loop would be 4187140 us. A NOP inserted at location "a" will add 20y us. A NOP at "b" will add 20xy us. The use of both NOPs will boost our maximum time to 5492740 us, or over 5 seconds.

The purpose of developing formulas is to determine the values of the registers needed to obtain a specified time period. For purposes of illustrating an example, let us assume we want exactly 5000 us to pass between point A and point B of a program. We would place a CAL instruction between point A and point B which would call the time loop. The shorter loop will be sufficient for this application, so the equation will now be

```
76X + 40 = 5000
```

Working the equation will give a value of 65.23615 . . . for "x." A fractional value will not fit in any single register of the CPU. To find out what to do now, multiply 65 by 76, add 40, and subtract the result from 5000. We find the difference

is 20 us. This is very simple to take care of. Insert a NOP instruction at any point in the routine where it will not be repeated. Before the LBI instruction would do fine. Now, with 65 (decimal notation) loaded into the B register, exactly 5000 us will pass between points A and B of our main program.

Finding an exact time period using the longer loop involves a certain amount of trial and error. To find an approximate value of "x" (using no NOPs) use this formula:

$$X = \left(\frac{N-40}{64Y}\right) - 1.5$$

Assign an arbitrary value to "y," replace "n" with the required time period.

Now, assume a time period of exactly 505904 us is needed. (This time period will be used later.) There is one stipulation in this case which will be explained in greater detail later. The value of "x" must be 255. Solving the formula equation for "y"

$$(64)(255)Y + 100Y + 40 = 505904$$

gives "y" a value of 30.8078. 30 must be used for "y." The total time of the loop is then 492640 us, 13264 us short of the required time. In most cases, you would re-assign values and try again, but in this case, the value of "x" cannot be changed. The alternative is to use the shorter loop to clean up the leftovers. After calling one loop, call the other loop. Then go on with the main program. Solving the short loop equation comes out at a nice even 174.

$$76X + 40 = 13264$$

What looked like a real oddball turned out to be perfect!

The formulas and all such may seem like a lot of monkey business just to waste time. Speed *is* the purpose of computers, but there are times when they must be slowed down.

The primary application of time loops is in I/O interface. If a computer is to monitor a data input which is to be read once every 10 ms, there are two alternatives for timing. The hardware of the device being monitored may include a timing device and a flag to indicate when the device is ready. The computer enters a loop which monitors the flag until the device is ready, then reads the data. The other alternative is to use the software time loop, and omit the extra hardware.

An interesting application along this line is a completely software "fabricated" keyboard debounce system. This method will not work in an interrupt type of input system, but for many small scale systems, this method is ideal.

Rather than connecting the *keypressed* line of the keyboard to some debounce, timer and latch circuitry, connect it to the eighth bit of the parallel data input on the computer. The loop used will test the eighth bit for the *keypressed* state. When a *keypressed* is sensed, a time loop of 16344 us is executed, then the data input is accepted. The loop then branches back to the main program to take care of the new data. When the program comes back to the input loop, the *keypressed* line is first tested to be sure no keys are being pressed. After all keys have been released, the loop will wait for the next *keypressed* state. This procedure will prevent more than one data entry from each keystroke.

When I first tried this keyboard debounce method over five months ago, I was so pleased with it that I'm still using the method for all data entry to my microcomputer. Not once has it missed some data, or given me false or duplicated data. And it was so easy to implement!

Fig. 3. Software can be used to debounce a keyboard — simply loop around for a long enough time to ensure that keys have stabilized. The loop is started as soon as the "any key pressed" (keypressed) line indicates any non-null bit pattern.

```
07/115 = 101   IN
   116 = 002   RLC    TEST INPUT FOR KEYPRESSED,
   117 = 100   JFC    WAIT UNTIL CONDITION IS
   120 = 115   L      SATISFIED
   121 = 007   H
   122 = 101   IN
   123 = 002   RLC    TEST INPUT FOR KEYPRESSED,
   124 = 140   JTC    WAIT UNTIL CONDITION IS
   125 = 122   L      SATISFIED
   126 = 007   H
   127 = 026   LCI
   130 = 377   "255"  EXECUTE
   131 = 021   DCC    TIME
   132 = 110   JFZ    DELAY
   133 = 131   L
   134 = 007   H
   135 = 101   IN     ACCEPT INPUT OF DATA
   136 = 104   JMP
   137 = 1     L      JUMP TO MAIN PROGRAM
   140 = H     H      (MAY BE REPLACED WITH A
                       RETURN INSTRUCTION.)
```

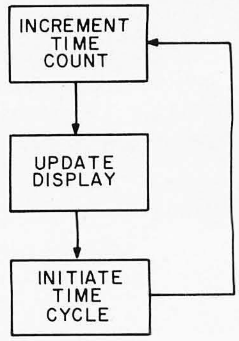


Fig. 4. The digital clock program looks simple at this level: Increment the time count, update the display, then initiate a time cycle such that the entire loop takes exactly one second!

Time loops may also be used in output applications. I have an SWTPC TV typewriter, but I am not using the special computer interface board. I found that a simple time loop does the job well enough and fast enough.

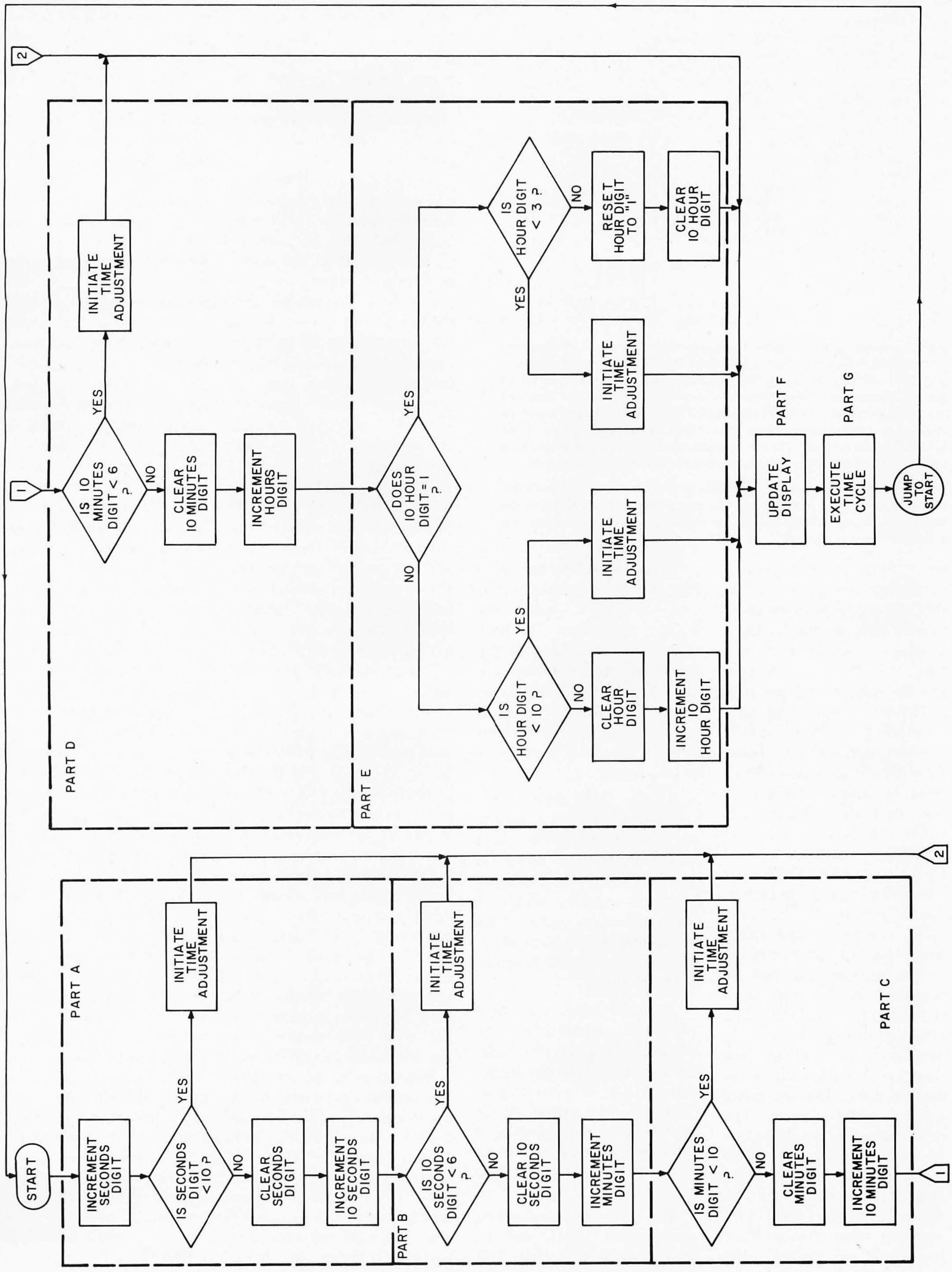
Since we've done our homework, now we can play. An interesting and novel application of time loops is a completely software "fabricated" clock. The clock program presented here will have three major functions (see Fig. 4).

The clock will display hours, minutes and seconds. The "increment time count" segment of the program is responsible for computing the next time reading in sequence. It must consist of more than a straight counting sequence since time is not expressed in straight decimal format.

The "update display" segment is responsible for producing the newly computed time at an output device.

After the first two segments have been executed,

Fig. 5. Ah, but the simplicity of Fig. 4 — as this figure reveals — hides a lot of low level detail. Here is the flow chart of the clock's operations.



8008 Timing Quick Reference Guide

US.	INSTRUCTION
20	INCREMENT INDEX REGISTER
20	DECREMENT INDEX REGISTER
20	ROTATE ACCUMULATOR
12/20	CONDITIONAL RETURN*
36/44	CONDITIONAL JUMP*
36/44	CONDITIONAL CALL*
20	UNCONDITIONAL RETURN
44	UNCONDITIONAL JUMP
44	UNCONDITIONAL CALL
20	RESTART
32	LOAD DATA IMMEDIATE (INTO INDEX REGISTER)
36	LOAD DATA IMMEDIATE (INTO MEMORY REGISTER)
32	ALU IMMEDIATE
20	ALU REGISTER
32	ALU MEMORY REGISTER
24	OUTPUT
32	INPUT
20	LOAD DATA - REGISTER(OP CODE 3--)
32	LOAD DATA - MEM. & REG. (OP CODE 3-7 OR 37-)

Here is a quick reference table for execution times of all instructions in the 8008 repertoire. Such a reference table can be easily made for any CPU. Simply multiply the number of machine states required for the execution of each type of instruction by the time required per machine state. AT 500 kHz, the 8008 takes four us per machine state. An unconditional jump instruction requires 11 states in the 8008 , therefore 44 us. Do not confuse machine states with machine cycles. The same jump instruction requires three machine cycles.

***Conditional instructions: Execution time depends upon condition. If condition causes true branch, the execution time is longer. If the condition causes a false branch (if condition is not satisfied), the execution time is shorter.**

the "time cycle" segment makes up the difference so that the entire program takes exactly one second per pass. Writing a clock program isn't hard, but making it take exactly one second per pass definitely adds to the challenge. The major consideration is that branches from conditional instructions must be balanced in such a way that the program will take exactly the same execution time regardless of the combination of conditions and branches. That's where all the time loops come in, and that's where lots of fun comes in!

The program can best be described in the form of a flowchart, Fig. 5. The program listing in Fig. 6 is divided according to the flowchart divisions shown by dashed lines. The op codes are for 8008 systems. The mnemonics and op codes can be easily translated into 8080 format. However, all timing considerations must be recalculated for use with anything other than an 8008 running at exactly 500 kHz.

When time balancing a segment of a program, it is best to work from the bottom and go up. The time adjustment in part A of the flowchart must compensate for parts B, C, D and E, so before that time period can be calculated, the execution time of the other parts must be calculated.

Some of the time adjustments in part E do not use a time loop. The short time adjustments there (in part E) are more conveniently implemented with a combination of other time consuming instructions which will not change the function of the program.

To determine the time adjustment needed in one branch, tabulate the total execution time of the longer branch. Add or subtract 8 us (depending upon which branch is the true branch) to compensate for the difference in conditional jump instructions.

The same time loop will be used several times, yet the time periods will vary. This can be accommodated when using the short loop by placing the LBI instruction and loading the value of "x" *before* the loop is called. The location of the LBI instruction will have no effect on the overall time period produced.

Occasionally a time loop will not come out evenly. For example, another 12 us may be needed. This will not be accommodated in the loop, so the only alternative is to use a NOP instruction. But the only instruction which will absorb just 12 us is an unsatisfied conditional return instruction. Using such an instruction could result in trouble if used alone. However, if an AND instruction can be used without affecting the program functions, the AND instruction will insure that the conditional return (RTC) will not be satisfied. To keep the program in balance when balancing the time, insert a NOP in the opposite branch to offset the AND instruction, and the net difference will be 12 us.

Flowchart parts F and G need not be included in the time balancing considerations of A, B, C, D and E. The program returns to a common point before executing parts F and G, so those parts are not offsetting anything.

The output loop as given in the listing will provide an ASCII output for a TV typewriter. A sufficient time loop is provided between each individual output operation. The output loop may be easily modified for use with other devices. For use with Teletype, a line feed command must be added to the output characters. (Only a carriage return is used with a TVT.) For use with an LED display, deleting the ORI instruction at location 04/257 will leave a straight binary (also BCD equivalent, since vales do not exceed 9) output. Keep in mind, however, that modifying any part of the program will also require modifying the timing elements involved.

The execution time of the complete "increment time count" segment plus the "update display" segment totals 494096 us. Subtract that time from one second to find the time required of the timing cycle. The required time is 505904 us. The values for this loop have already been worked out in a previous example.

The reason the value of "x" cannot be conveniently changed in the long loop in this case is that the loop is called and used from two locations in the program. The value of "x" cannot be changed for one application without affecting the other. If the loop were modified to load B from another register which remained constant, both values would become variables which could be easily assigned values from any point in the program. This would also include recalculating the time formula of the loop.

Your clock should now be ready to run. (Oh, by the way, there is one little drawback: Your computer can't be used for anything else while it's keeping time, unless, of course, you really want to go to extremes with the calculating! This program is strictly a novelty!) When you are ready to start your clock, load the correct time plus a couple of minutes into memory locations 04/000 through 04/005. When the loaded time comes, start the computer. Jump into the program at 04/006.

The time kept by the computer will only be as accurate as the frequency of the clock driving the CPU. The oscillator must be set at exactly 500 kHz. Although this is difficult to do, any percentage of error in frequency will be directly reflected by the time kept. ■

Fig. 6. And finally, the lowest level of detail of all: A "pseudo assembly" listing of the program for the digital clock as implemented for an 8008 computer. Of course, those readers who have an 8080, a 6501, a 6800 or PACE will have to do a little bit of thinking to adapt Fig. 4 and Fig. 5 to the alternative microcomputer CPU designs.

```
  THOUR:004/000 = XXX            10 HOUR DIGIT REGISTER
   HOUR:004/001 = XXX            HOUR DIGIT REGISTER
   TMIN:004/002 = XXX            10 MINUTE REGISTER
    MIN:004/003 = XXX            MINUTE REGISTER
   TSEC:004/004 = XXX            10 SECOND REGISTER
    SEC:004/005 = XXX            SECOND REGISTER
- - - - - - -
  START:004/006 = 056  LHI       LOAD L/H WITH
        004/007 = 004  H(SEC)    ADDRESS OF SECONDS
(A)     004/010 = 066  LLI       DIGIT REGISTER
        004/011 = 005  L(SEC)
        004/012 = 307  LAM       INCREMENT SECONDS
        004/013 = 004  ADI       DIGIT
        004/014 = 001  "1"
        004/015 = 074  CPI       DECISION: JUMP IF
        004/016 = 012  "10"      SECONDS DIGIT IS NOT
        004/017 = 100  JFC       LESS THAN 10
        004/020 = 036  L ─┐
        004/021 = 004  H ─┘──► GTENS
        004/022 = 370  LMA       RETURN SECONDS DIGIT TO ITS REGISTER
        004/023 = 370  LMA       REPEAT INSTRUCTION FOR MORE TIME
        004/024 = 016  LBI
        004/025 = 017  "15"      SET VALUE OF "X"
        004/026 = 106  CAL       CALL TIME LOOP TO COMPENSATE
        004/027 = 247  L ─┐
        004/030 = 004  H ─┘──► TLOOP
        004/031 = 300  NOP       NEED A LITTLE MORE TIME
        004/032 = 300  NOP
        004/033 = 104  JMP       FINISHED THIS CYCLE
        004/034 = 275  L ─┐
        004/035 = 004  H ─┘──► DISPL
  GTENS:004/036 = 006  LAI
        004/037 = 000  "0"       CLEAR SECONDS DIGIT REGISTER
        004/040 = 370  LMA
        004/041 = 061  DCL
        004/042 = 307  LAM       INCREMENT
        004/043 = 004  ADI       10 SECONDS DIGIT
        004/044 = 001  "1"
- - - - - - -
(B)     004/045 = 074  CPI       DECISION: JUMP IF
        004/046 = 006  "6"       10 SECONDS DIGIT IS NOT
        004/047 = 100  JFC       LESS THAN 6
        004/050 = 063  L ─┐
        004/051 = 004  H ─┘──► GOMIN
        004/052 = 370  LMA       RETURN 10 SEC. DIGIT TO REGISTER
        004/053 = 016  LBI
        004/054 = 015  "13"      SET VALUE OF "X"
        004/055 = 106  CAL       CALL TIME LOOP
        004/056 = 247  L ─┐
        004/057 = 004  H ─┘──► TLOOP
        004/060 = 104  JMP       FINISHED THIS CYCLE
        004/061 = 275  L ─┐
        004/062 = 004  H ─┘──► DISPL
 GOMIN: 004/063 = 076  LMI       CLEAR 10 SEC. DIGIT REGISTER
        004/064 = 000  "0"       (LAI, LMA ARE USED INSTEAD OF LMI
        004/065 = 061  DCL       WHERE TIMING WORKS OUT BETTER THAT WAY.)
        004/066 = 307  LAM       INCREMENT
        004/067 = 004  ADI       MINUTES DIGIT
        004/070 = 001  "1"
- - - - - - -
        004/071 = 074  CPI       DECISION: JUMP IF
(C)     004/072 = 012  "10"      MINUTES DIGIT IS NOT
        004/073 = 100  JFC       LESS THAN 10
        004/074 = 110  L ─┐
        004/075 = 004  H ─┘──► GTENM
        004/076 = 370  LMA       RETURN MINUTES DIGIT TO REGISTER
        004/077 = 016  LBI
        004/100 = 012  "10"      SET VALUE OF "X"
        004/101 = 106  CAL       CALL TIME LOOP
        004/102 = 247  L ─┐
        004/103 = 004  H ─┘──► TLOOP
        004/104 = 317  LBM       NEED AN EXTRA 12 US.
        004/105 = 104  JMP       (LBM - 32 US, OTHER 20 US  BALANCED BY NOP)
        004/106 = 275  L ─┐
        004/107 = 004  H ─┘──► DISPL
 GTENM: 004/110 = 300  NOP       20 US BALANCE
        004/111 = 006  LAI
        004/112 = 000  "0"       CLEAR MINUTES REGISTER
        004/113 = 370  LMA
        004/114 = 061  DCL
        004/115 = 307  LAM       INCREMENT
        004/116 = 004  ADI       10 MINUTES DIGIT
        004/117 = 001  "1"
- - - - - - -
(D)     004/120 = 074  CPI       DECISION: JUMP IF
        004/121 = 006  "6"       10 MINUTES DIGIT IS NOT
        004/122 = 100  JFC       LESS THAN 6
        004/123 = 137  L ─┐
        004/124 = 004  H ─┘──► GOHOUR
        004/125 = 370  LMA       RETURN 10 MIN. DIGIT TO REGISTER
        004/126 = 016  LBI
        004/127 = 007  "7"       SET VALUE OF "X"
        004/130 = 106  CAL       CALL TIME LOOP
        004/131 = 247  L ─┐
        004/132 = 004  H ─┘──► TLOOP
        004/133 = 300  NOP       KEEPING THE TIME IN BALANCE
        004/134 = 104  JMP       FINISHED THIS CYCLE
        004/135 = 275  L ─┐
        004/136 = 004  H ─┘──► DISPL
GOHOUR: 004/137 = 006  LAI
        004/140 = 000  "0"       CLEAR 10 MINUTES REGISTER
        004/141 = 370  LMA
        004/142 = 061  DCL
        004/143 = 307  LAM       INCREMENT
        004/144 = 004  ADI       HOURS DIGIT
        004/145 = 001  "1"
- - - - 004/146 = 370  LMA       PUT THE HOURS DIGIT BACK FOR THE
        004/147 = 061  DCL       TIME BEING.
(E)     004/150 = 307  LAM
        004/151 = 074  CPI       DECISION: JUMP IF
        004/152 = 001  "1"       10 HOUR DIGIT = 1
        004/153 = 150  JTZ
        004/154 = 213  L ─┐
        004/155 = 004  H ─┘──► NOON
        004/156 = 060  INL
        004/157 = 307  LAM
        004/160 = 074  CPI       DECISION: JUMP IF
        004/161 = 012  "10"      HOUR DIGIT IS NOT
        004/162 = 100  JFC       LESS THAN 10
        004/163 = 177  L ─┐
        004/164 = 004  H ─┘──► GTENH
        004/165 = 307  LAM       NEED      ***A NOP IS ADDED TO BOTH BRANCHES
        004/166 = 307  LAM       TO        TO BALANCE  THE TRUE  BRANCH FROM
        004/167 = 307  LAM       WASTE     04/153. A BETTER PLACE FOR THE NOP
        004/170 = 307  LAM       200 US    WOULD HAVE BEEN  04/156,  BUT WHO
        004/171 = 307  LAM       TO        WANTS TO REWRITE HALF A PROGRAM TO
        004/172 = 300  NOP       BALANCE   SAVE ONE MEMORY LOCATION?
        004/173 = 300  NOP       BRANCH
        004/174 = 104  JMP       NOW LET'S GET OUT'A HERE
        004/175 = 275  L ─┐
        004/176 = 004  H ─┘──► DISPL
 GTENH: 004/177 = 006  LAI
        004/200 = 000  "0"       CLEAR HOUR DIGIT REGISTER
        004/201 = 370  LMA
        004/202 = 061  DCL
        004/203 = 307  LAM       INCREMENT
        004/204 = 004  ADI       10 HOUR DIGIT
        004/205 = 001  "1"
        004/206 = 370  LMA       RETURN 10 HR. DIGIT TO REGISTER
        004/207 = 300  NOP       ***
        004/210 = 104  JMP       CYCLE FINISHED
        004/211 = 275  L ─┐
        004/212 = 004  H ─┘──► DISPL
  NOON: 004/213 = 060  INL
        004/214 = 307  LAM
        004/215 = 074  CPI       DECISION: JUMP IF
        004/216 = 003  "3"       HOUR DIGIT IS NOT
        004/217 = 100  JFC       LESS THAN 3
        004/220 = 232  L ─┐
        004/221 = 004  H ─┘──► RESHOUR
        004/222 = 016  LBI
        004/223 = 002  "X"       SET VALUE OF "X"
        004/224 = 106  CAL       CALL TIME LOOP
        004/225 = 247  L ─┐
        004/226 = 004  H ─┘──► TLOOP
        004/227 = 104  JMP       JUMP TO 004/275/036
        004/230 = 275  L ─┐
        004/231 = 004  H ─┘──► DISPL
RESHOUR:004/232 = 006  LAI
        004/233 = 001  "1"       RESET HOUR DIGIT TO "1"
        004/234 = 370  LMA
        004/235 = 061  DCL
        004/236 = 006  LAI
        004/237 = 000  "0"       CLEAR 10 HOUR DIGIT REGISTER
        004/240 = 370  LMA
        004/241 = 241  NDB       (YES, 241 = 241; THAT'S NOT AN ERROR)
        004/242 = 043  RTC       FOR TIME BALANCING. THE NET DIFFERENCE
        004/243 = 043  RTC       BETWEEN BRANCHES FROM 04/153 WAS 24 US.
        004/244 = 104  JMP       2 X RTC = 24 US. THE NDB IS BALANCED
        004/245 = 275  L ─┐                 BY THE NOP'S MENTIONED
- - - - 004/246 = 004  H ─┘──► DISPL        IN THE NOTE ***
  TLOOP:004/247 = 011  DCB       SHORT TIMING LOOP
        004/250 = 053  RTZ
        004/251 = 104  JMP
        004/252 = 247  L ─┐
        004/253 = 004  H ─┘──► TLOOP

   OUTL:004/256 = 307  LAM       OUTPUT SUBROUTINE STARTS
        004/257 = 064  ORI       GENERATE ASCII CHARACTER
        004/260 = 060  "48"
        004/261 = 121  OUT       PRINT CHARACTER
        004/262 = 026  LCI
        004/263 = 005  "5"       SET VALUE OF "Y"
        004/264 = 106  CAL       CALL LONG TIME LOOP
        004/265 = 324  L ─┐
        004/266 = 004  H ─┘──► LTIME
        004/267 = 031  DCD
        004/270 = 053  RTZ       ARE WE DONE PRINTING?
        004/271 = 060  INL       CONTINUE IF NOT
        004/272 = 104  JMP
        004/273 = 256  L ─┐
        004/274 = 004  H ─┘──► OUTL
- - - - - - -
  DISPL:004/275 = 036  LDI       SET UP COUNT - DISPLAY ROUTINE
        004/276 = 006  "6"
(F)     004/277 = 066  LLI       SET UP ADDRESS
        004/300 = 000  "0"
        004/301 = 106  CAL       OUTPUT ROUTINE
        004/302 = 256  L ─┐
        004/303 = 004  H ─┘──► OUTL
        004/304 = 006  LAI
        004/305 = 015  "13"      OUTPUT CARRIAGE RETURN COMMAND
- - - - 004/306 = 121  OUT
        004/307 = 026  LCI
(G)     004/310 = 036  "30"      SET VALUE OF "Y"
        004/311 = 106  CAL       CALL LONG TIME LOOP
        004/312 = 324  L ─┐
        004/313 = 004  H ─┘──► LTIME
        004/314 = 016  LBI
        004/315 = 255  "175"     SET VALUE OF "X"
        004/316 = 106  CAL       CALL SHORT TIME LOOP
        004/317 = 247  L ─┐
        004/320 = 004  H ─┘──► TLOOP
        004/321 = 104  JMP       JUMP BACK TO THE BEGINNING AND RECYCLE
        004/322 = 006  L ─┐
- - - - 004/323 = 004  H ─┘──► START
  LTIME:004/324 = 016  LBI       SET VALUE OF "X"
        004/325 = 377  "255"
  LTIM1:004/326 = 011  DCB       DECREMENT "X"
        004/327 = 110  JFZ       JUMP BACK TO DECREMENT AGAIN
        004/330 = 326  L ─┐
        004/331 = 004  H ─┘──► LTIM1
                                 IF "X" DOESN'T EQUAL "0"
        004/332 = 021  DCC       DECREMENT "Y"
        004/333 = 053  RTZ       GO BACK TO PROGRAM IF Y = 0
        004/334 = 104  JMP       REPEAT LOOP
        004/335 = 324  L ─┐
        004/336 = 004  H ─┘──► LTIME
        004/END = SET THE TIME A MINUTE OR TWO IN ADVANCE AT LOCATIONS 04/005,
                  WAIT UNTIL THE RIGHT TIME, AND START THE PROGRAM BY JUMPING
                  IN  AT 04/006.  THE PROGRAM  IS STOPPED BY  LOADING A  HALT
                  INSTRUCTION INTERRUPT FROM THE FRONT PANEL.
```

BYTE'S BITS

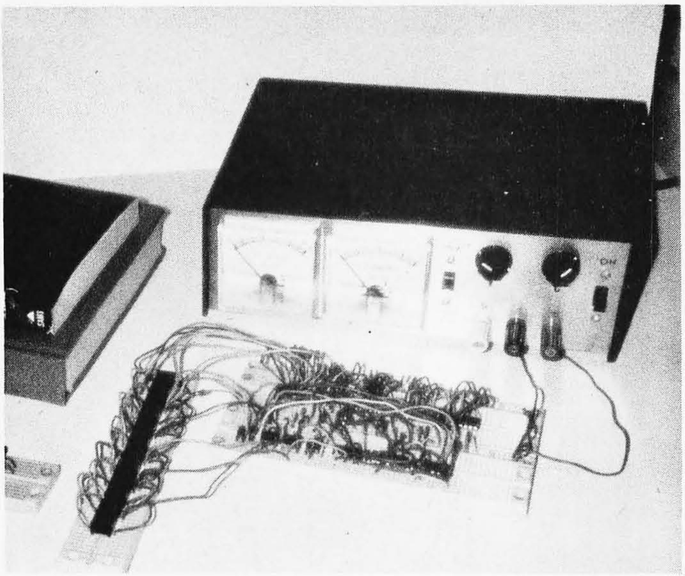
Getting Started in TTL

Digital logic is one of the many fascinating aspects of electronics. To date, TTL logic is the heart of almost every home brew computer system. Programming a computer is an exciting exploration into a mysterious world. Exploring the world of hardware that makes the programming possible is quite another exciting adventure. The best way to explore either of these worlds is not just through reading manufacturers data, but through experimenting for yourself.

A TTL integrated circuit is a simple device by itself. When combined with a number of devices, a very complex system can evolve. Don't let the complexity of a computer system get you down. You don't need to be able to design computer systems to start having fun with TTL. Nor do you need be an electronic engineer. You don't need to know how to build automobile engines to get your car started, do you?

To get started, you need only a few items. You will need a +5 volt power supply rated at 1 Amp or so. TTL integrated circuits do not require anything near 1 Amp each, but a number of ICs used together may add up to more than 1 Amp.

The most convenient method for mounting and connecting integrated circuits is the use of a solderless terminal strip, the type advertised in BYTE and many other electronics publications. Only a little wire (22 or 24 gauge wire) is needed for interconnecting the TTL devices.



Exploring the world of hardware with a dc power supply and a solderless terminal strip.

A good starting selection of TTL integrated circuits might include the following.

quantity	type	description
2	7400	Quad 2-input pos. NAND gate
2	7404	Hex inverter
2	7408	Quad 2-input pos. AND gate
2	7410	Triple 3-input pos. NAND gate
2	7411	Triple 3-input pos. AND gate
1	7420	Dual 4-input pos. NAND gate
1	7421	Dual 4-input pos. AND gate
1	7473	Dual J-K flip flop

The 7400 series part numbers are standard. Different manufacturers may add one or two letters to the number, but it is still the same part.

A TTL device has only three types of connections: Input, output and power supply terminals. The power supply terminals will be ground (GND) and Vcc. The voltage applied to Vcc should be +5 for proper operation.

The inputs and outputs will always be in one of two states: HIGH or LOW. In the HIGH state, the voltage will be over 2.4 volts. In the LOW state, the voltage will be under 0.4 volts. Only a simple LED is needed for a state indicator. The circuit design physics are simple: Connect the output of one device directly to the input of another device, provided the TTL ICs used are not the open collector type. The ICs listed above are not the open collector type.

Your next most important item will be your literature. There are many details not discussed here which you will want to learn as you advance in TTL designing. There have been many books written on the subject of TTL. One very good book for hobbyists is Don Lancaster's *TTL Cookbook*, which provides much information valuable to newcomers to the field, yet is detailed enough to be of great value even after you are quite familiar with TTL.

A second source of information is a TTL data book, available from the manufacturers. Most TTL suppliers will supply data sheets for free or for a nominal fee. A data book is a complete collection of data sheets for all TTL manufactured by the company you obtained the book from. The data sheets or book will show the logic diagram, actual circuit diagram, pin connections, and circuit characteristics and functions for each TTL device.

Your own experience should be proof that TTL is no more difficult than any other aspect of computers. So start planning your expedition!

REFERENCE BOOKS ON TTL

Texas Instruments Inc.
PO Box 3640 M/S 84
Dallas TX 75285

TI supplies the following excellent references. To order send a personal check or money order to the above address.

LCC4200 Semiconductor Memory Data Book, *$2.95.*

LCC4151 Linear & Interface Circuits Data Book, *$3.95.*

LCC4111 TTL Data Book, *$3.95.*

LCC4161 Supplement to the TTL Data Book, *$1.95.*

TTL Cookbook, *by Don Lancaster. Howard W. Sams & Co., Indianapolis, Indiana, 1974. $8.95. See the review in BYTE #1, page 85, for details about this excellent introductory text and reference source.*

Use of ASCII Approved for Amateur Satellites

The FCC has issued a Special Temporary Authority (STA) to the Radio Amateur Satellite Corporation (AMSAT) allowing the use of ASCII by radio amateurs through the communications packages aboard the OSCAR 6 and OSCAR 7 satellites. The STA has been granted until Feb. 28, 1976. At the conclusion of this period, AMSAT will compile a report of the results of the experiments conducted and callsigns of the amateurs involved. More information can be obtained from AMSAT, Box 27, Washington DC 20044 phone (202) 488-8649.

Gary L. Tater W3HUC
7925 Nottingham Way
Ellicott City MD 21043

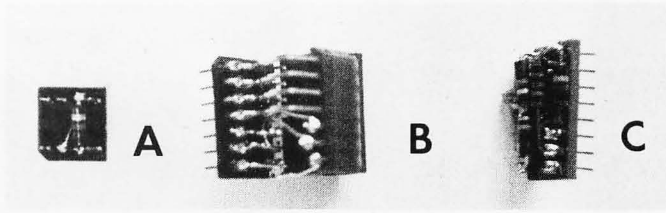


Micro-440

Computers come in all sizes. Here is a product made by COMP-SULTANTS Inc., PO Box 1016, Huntsville AL 35807. The product is the Micro-440 computer, based upon the Intel 4040 chip, and is to my knowledge the first such kit available. The photo shows an assembled version of the product, which costs $375 including 256 bytes of RAM program memory, power supply, cabinetry and displays. The entire CPU is constructed on one PC board. A kit version is available at $275, and COMP-SULTANTS will sell partial kits as well.

Information can be had for 25¢ in coin, or you can order the user's manual (440-U) for $10, which is refundable with an order for the product.

The Intel 4040 instruction set operates upon 4-bit words in scratch pad register areas, and will prove adequate for many of the simpler programming applications readers might contemplate. It will prove to be an excellent learning machine at a fairly reasonable price.



Creative Uses of DIP Header Plugs

Sumner S. Loomis (Loomis Laboratories, Box 131-A, Prairie Point MS 39353) sends along three different examples of DIP header plugs used to mount components in existing circuitry. The first, A, is a plug-in substitute for an operational amplifier — a simple resistor used in an existing circuit which did not need the op amp. B is an example of how a socket can be attached to a DIP header plug in order to partially rewire a circuit without touching the original wiring. Several of the socket pins are bent and connected directly to the same pins of the plug. Other socket pins are clipped and rewired to different pins of the plug. By saving the original wiring, this technique leaves you the option of going back to the original IC — or you can use this technique to correct for errors in a PC board. The example of C shows several resistors mounted on the plug. This technique of mounting small discretes on plugs enables you to use sockets for these parts. It saves the hassle of installing individual wrapping posts.

Notes

As BYTE #4 goes to press, *Electronic News* (Sept. 29, 1975) is running an advertisement for the Advance Micro Devices version of an 8080 design: The Am9080A. Significant is that the 100 quantity price of this chip is now down to $29.95. Price drops go even further than that — if you want to buy a million of the processors, you'll get a real bargain at $6 each.

This is further evidence that the cost of silicon in a home computer is getting closer and closer to the cost of the iron used to contain it.

* * *

" . . . But the roots of the software misery go deeper. It comes from the fact that people are forced to live with machines that they do not want. They have not constructed them, they simply receive them and have to make the best out of it . . . Thus, software engineering, for the time being, is partly a defense stratagem. But, I hope that some day this situation will turn around, I hope one day software engineering considerations will dictate how machines are to be built and then to be used . . ."

Friedrich L. Bauer in the Preface to
Software Engineering, An Advanced Course,
Springer-Verlag Publishers, 1973.

* * *

"A lot of comparisons have been made about how much faster computers can do things than humans can. Here is another aspect of this comparison: To make all the errors a modern computer can make in 10 seconds, it would take a team of 30 accountants working full-time for 50 years."

Barry W. Boehm, Guest Editor,
IEEE Transactions on Software Engineering,
Vol. SE-1, No. 2 (June 1975), p. 137.

Try to Duplicate This Using Magnetic Tape!

William D. Roch (5133 Catalon Ave., Woodland Hills CA 91364) sends in this sample of an ASCII character set translated into codes which will punch real dot matrix letters on paper tape. It provides a way for people with paper tape equipment to put a human-readable leader onto a tape prior to punching the usual data.

In order to conveniently use this method, you'll have to make a table in memory to contain the full 6-bit ASCII subset shown in the table. Then you'll have to write a little routine to take a *character string* input parameter, look up each character in the table, and output the appropriate sequence of codes. The table lists what looks like gibberish, but when you punch out the sequences shown, you will see character patterns in the holes on the tape.

```
A = @>II>@        SPACE = @@@@@@
B = @?%%Z@            ! = @W@
C = @^!!R@            " = @G@G@
D = @?!!^@            # = @J?J?J@
E = @?%%!@            $ = @R%?)P@
F = @?EEA@            % = @SK42@
G = @^!):H@           & = @^-4(@
H = @?DD?@            ' = @G@
I = @!?!@             ( = @^!@
J = @P  ?@            ) = @!^@
K = @?LR!@            * = @RL?LR@
L = @?  @             = = @LLL@
M = @?BDB?@           - = @DDD@
N = @?BDH?@           @ = @\"-.P@
O = @?!!?@            [ = @?!@
P = @?IIF@            \ = @ADH @
Q = @^!1>@            + = @$.$@
R = @?IY&@            ; = @^>@
S = @V+)Z@            ^ = @$B?B$@
T = @AA?AA@           ] = @!?@
U = @?  ?@            < = @LR!@
V = @OP PO@           > = @!RL@
W = @?PHP?@           , = @X8@
X = @!RLR!@           . = @XX@
Y = @AB<BA@           ? = @BA)F@
Z = @1)%#@            / = @ HDA@

          1 = @!? @
          2 = @9%%"@
          3 = @!%%Z@
          4 = @HLJ?H@
          5 = @W%-Y@
          6 = @^%%X@
          7 = @AA9G@
          8 = @Z%%Z@
          9 = @F))^@
          0 = @^!!^@
```

BYTE'S BITS

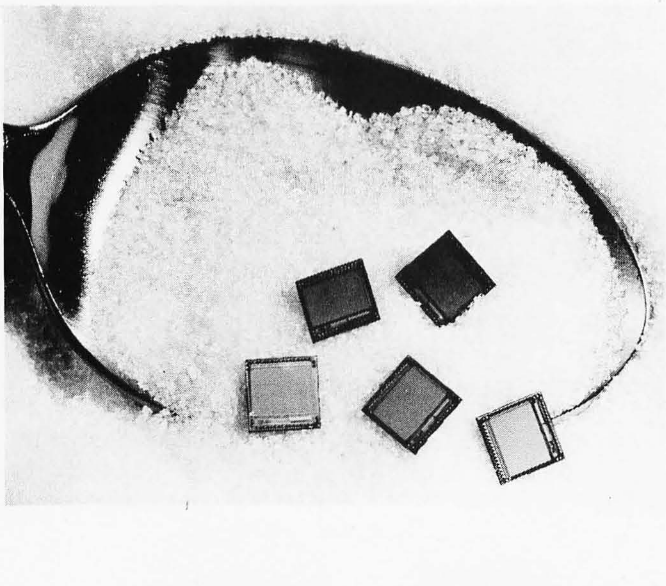


Welcome, IBM, to personal computing

With the announcement of the IBM 5100 system in a press release dated Sept. 9, 1975, personal computing gains an entry from the industry's production and service giant, IBM. The IBM 5100 is being marketed primarily as a *problem solver* for industrial, commercial and professional people – with the result that it is a very professional package at a premium price. But you will get a lot of function when you buy one of these computers – and you'll be able to call upon IBM's longstanding reputation for good service and customer handholding, the points which have led to the commendable success of IBM as a computer company.

What IBM engineers have done is to design a 50 lb-package of interactive personal computing which includes the following major features as standard items:

- System software is built-in, with access to BASIC and/or APL depending upon options purchased. These languages and the necessary monitor programs are hardwired into a read only memory.
- A video screen is built-in, with up to 1024 characters displayed in a 16-line by 64-character format.
- An interactive keyboard is standard, including the usual text entry section as well as a separate calculator style keypad. The keyboard has special function coding for all the APL and BASIC syntax elements.
- User memory starts at 16K bytes in the minimum configuration and can be expanded to 64K bytes (65,536).
- A magnetic tape cartridge storage device is standard. This is built into the unit, and becomes the primary method of storing user data and programs. It is also used to load IBM supplied programming packages. The cartridges for this device hold up to 204,000 characters of information.



You get all this function and professionalism from IBM by paying a high price. This machine is not intended to be a toy, although it would make an excellent one. It is intended as a production tool for people who presently use time sharing terminals, programmable calculators or other personal computers in daily work. Prices mentioned in the press release are:

- IBM 5100 processor ... $8975 to $19,975, depending upon user memory (16K, 32K, 48K or 64K bytes) and language (APL or BASIC or both) options.
- IBM 5103 printer ... $3,675 purchases an 80 cps 132-column dot matrix line printer.
- IBM 5106 Auxiliary tape unit ... $2,300 purchases an additional tape cartridge drive to augment the functions of the built-in drive.
- "Problem Solver Library" software is available for a one time *rental* of $500 including a wide range of utility and applications software with interactive user sequences.

Miscellaneous features also available for the machine include a TV monitor output, the external I/O adaptor used with the 5103 and 5106 devices, a communications adaptor which makes the 5100 emulate an IBM 2741 communications terminal, and a carrying case.

As an IBM engineered product, you can expect a solidly built computer. If you are a business or professional person needing a high quality calculational and programming tool, then you should investigate the 5100 as an item of capital equipment – which you can incidentally use to program numerous BASIC games when you're not using it for business. But if your sole interest in the machine is as a luxury toy, you have to be moderately well off to purchase the IBM 5100 at its present price. ■

THERE'S SOMEONE WHO WORKS FOR YOU WHO HAS CANCER. AND DOESN'T KNOW IT. HELP FIND HIM.



You don't know who he is. He doesn't know who he is. But there is a way to find him.

By letting us help strengthen your employee health program with our Employee Education Program. The purpose of the program is simple. We want to save lives by exposing your employees to sound facts and recommendations for action about cancer.

We'll supply free films, exhibits, speakers, pamphlets, posters, articles for your company publications.

We'll help you arrange "action" programs for your employees...clinics to help cigarette smokers quit, instruction in breast self-examination, screenings for cervical cancer via the Pap test. All with one purpose: to educate your people about the lifesaving facts of cancer.

Because if cancer is detected in its early stages, chances for cure are markedly increased.

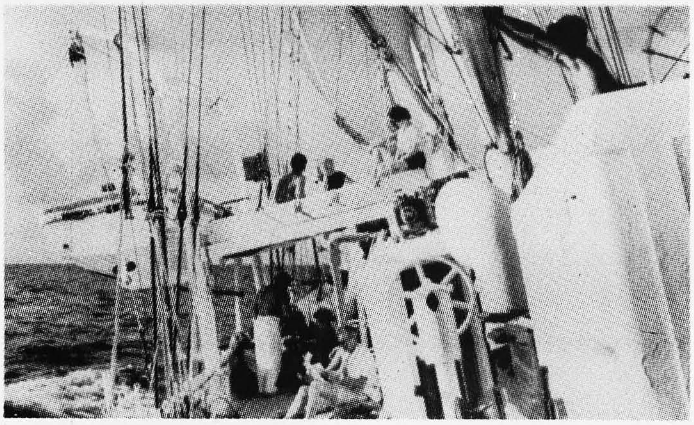
For more information contact your local American Cancer Society Unit.

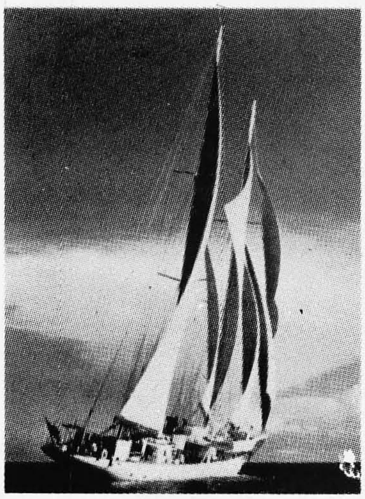
You will have helped to save someone's life.

Maybe, even your own.

AMERICAN CANCER SOCIETY

Take off your shoes.







Hit the deck in shorts and a tee shirt. Or your bikini if you want.

You're on a leisurely cruise to remote islands. With names like Martinique, Grenada, Guadeloupe. Those are the ones you've heard of.

A big, beautiful sailing vessel glides from one breathtaking Caribbean jewel to another. And you're aboard, having the time of your life with an intimate group of lively, fun-loving people. Singles and couples, too. There's good food, "grog," and a few pleasant comforts...but there's little resemblance to a stay at a fancy hotel, and you'll be happy about that.

Spend ten days exploring paradise and getting to know congenial people. There's no other vacation like it.

Your share from $245. A new cruise is forming now. Write Cap'n Mike for your free adventure booklet in full color.

Windjammer Cruises.
A WINDJAMMER INTERNATIONAL SUBSIDIARY · OTC

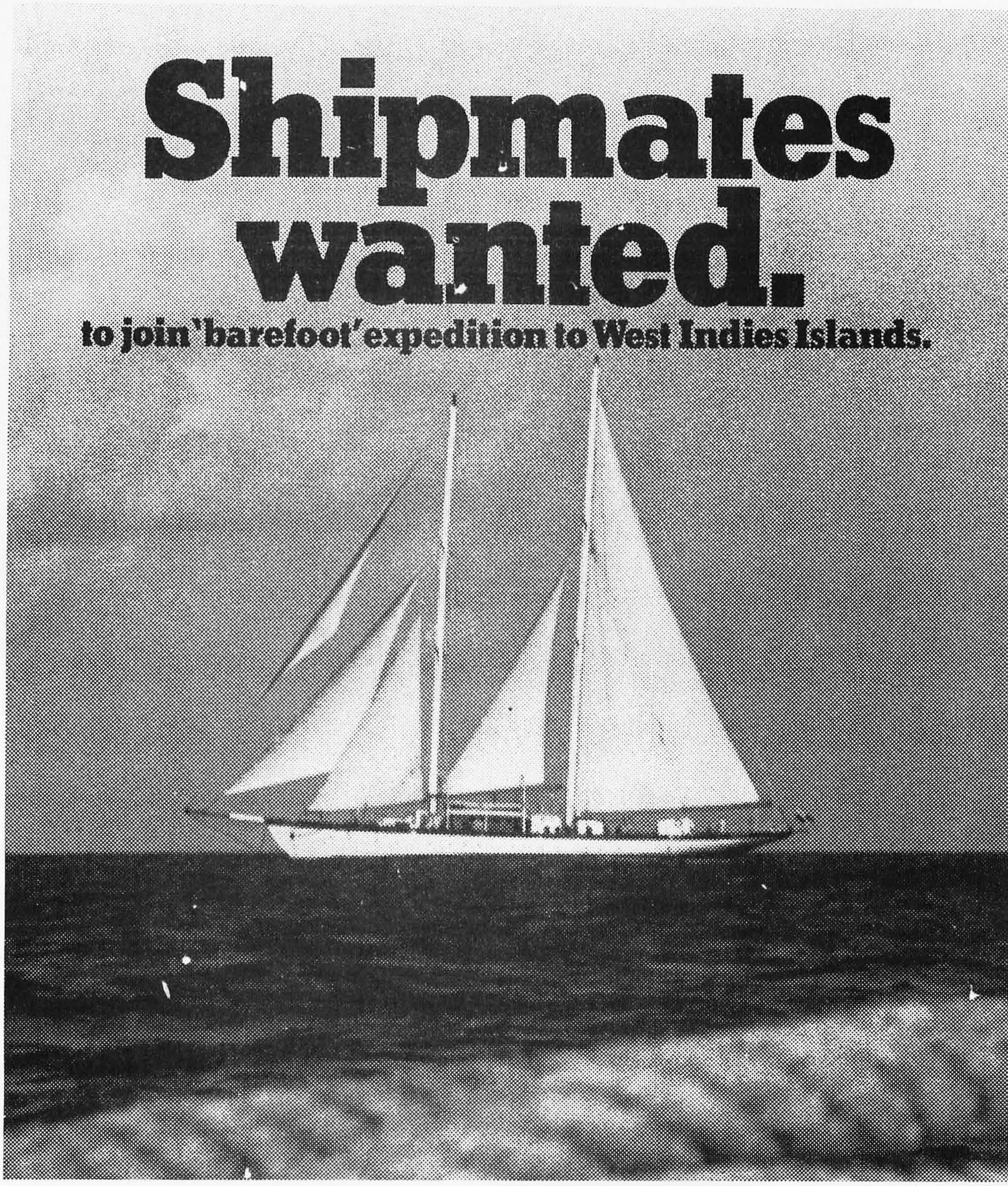
Name ____________________

Address ____________________

City ____________ State ____________ Zip ____________

Phone ____________________

P.O. Box 120, Dept. 121 Miami Beach, Florida 33139



And, we'll promise you only one thing. The most adventurous vacation you have ever experienced. Hop aboard our schooner for 10 sunfilled days and moonlit nights. A congenial group of shipmates now forming and setting their heading for Saba, Grenada, St. Lucia, Guadaloupe. And, Dutch St. Maarten. And, French Martinique. Explore pink, white and black sand beaches. Scale the cliffs, climb a volcano, prowl old forts and quaint towns. Hoist the sails, take a turn at the wheel or feet on the rail and let us do the work. Then limbo to a steel band and fall asleep under a star spangled Caribbean sky. Share from $250. If you have the spirit...come Windjammin'... a true life adventure. This coupon will bring you a full color adventure brochure.

Send me your color brochure on 10 day 'barefoot' vacations from $250.

Name ______________________________

Address ______________________________

City ______________ State ______________ Zip ______

Windjammer Cruises.

Post Office Box 120, Dept. 00, Miami Beach, Florida 33139.

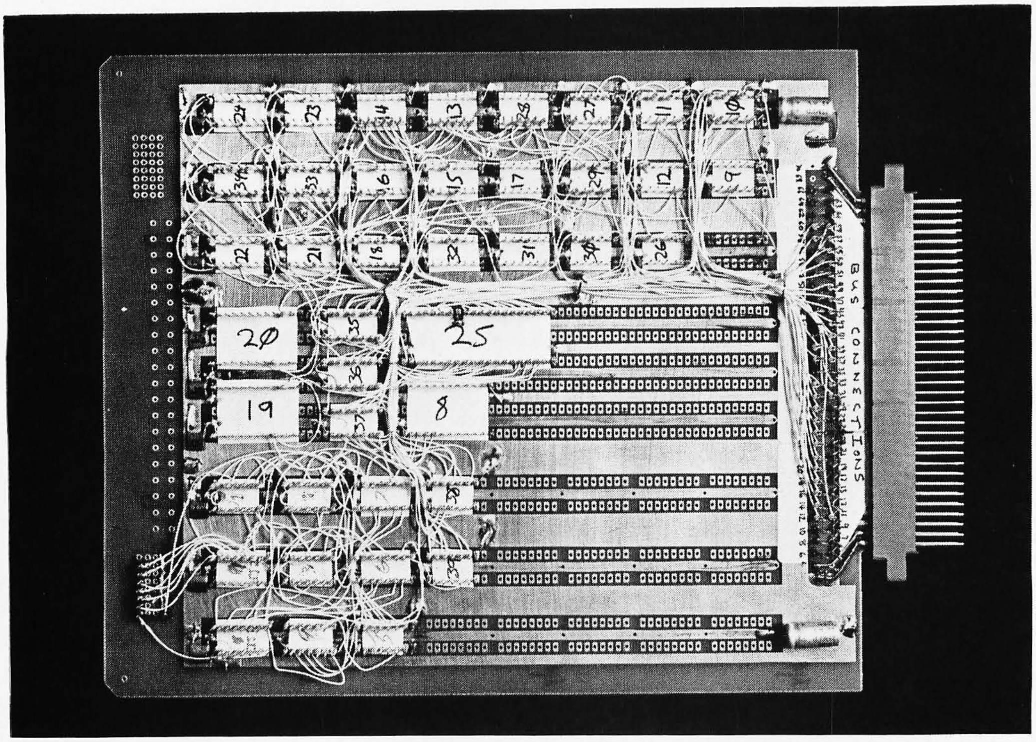


Fig. 1. One of the most convenient but permanent methods of assembly for experimental computer systems and logic designs is wire wrap construction using sockets and a general purpose prototyping board. This photograph illustrates an overall view of a partially wired Motorola 6800 computer system fabricated on a board available from CELDAT Design Associates, Box 752, Amherst, N.H. This board provides a basic socket mounting matrix for 14 and 16 pin sockets plus a general purpose area in the center which will accept bigger sockets.

The board which was photographed for this photo essay is the prototype Motorola 6800 system which is being designed and constructed for the LIFE Line application.

Photographic Notes on Prototype Construction

by
Carl Helmers
Editor, BYTE

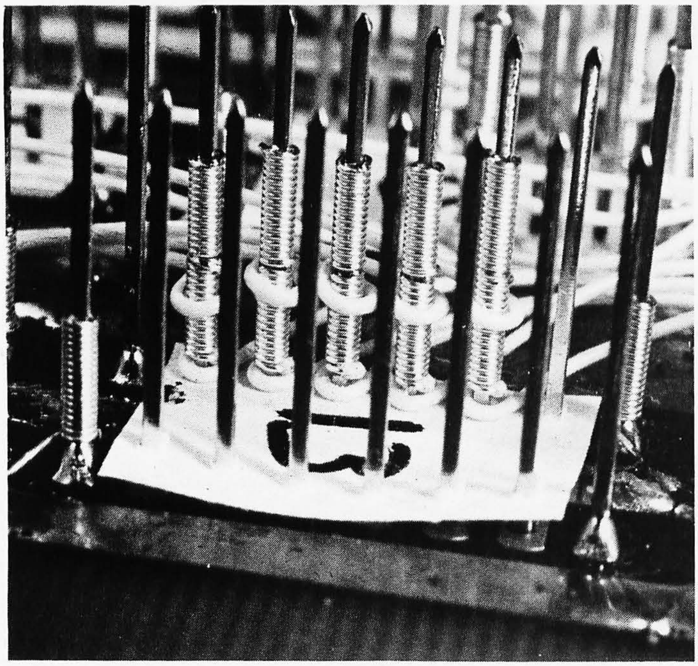


Fig. 2. As photographer Ed Crabtree zooms in for a closer look, we see details of several useful techniques. These techniques can be used with any one of several brands of general purpose boards which are available from manufacturers.

Sockets can be mounted by carefully soldering the four corner pins of the pads in the socket patterns of the general purpose board.

In the typical general purpose board, power and ground distribution is provided by alternate printed circuit strips on the wiring side of the board. Short jumper wires are wrapped to the socket pin and soldered to the appropriate bus as shown by the examples in the foreground.

Before wiring, a sheet of paper can be punched with a socket pattern using a spare socket and some perforated board. After cutting the paper to size and writing an identifying number for the circuit, the resulting label is slipped over the pins.

Several examples of two-level wire wrap interconnections can be seen in this photograph. The typical wire wrap post can accommodate three levels but good practice limits the use to two levels as a general rule.



SHARE BYTE

The next time a friend wants to borrow your BYTE sit right down and buy him a subscription and get him off your back. You know darned well that he isn't going to give you back your magazine, so be practical about it.

Subscriptions are only $12 – cheap enough to spread joy and fun among a few friends – like for Christmas. Tell you what – the second and onward subscriptions will only be $10 – Christmas Special – good until the end of the year.

A subscription makes a great Christmas present because the recipient gets a reminder every month for a year of your thoughtfulness.

Be a saint.

Cut

Name ....

Address ....

City .... State .... Zip ....

☐ BILL ME ☐ Check for $12 enclosed
☐ Bill BankAmericard or MasterCharge # ....

GIFT — Tear

Name ....

Address ....

City .... State .... Zip ....

☐ BILL ME ☐ Check for $10 enclosed

GIFT — Shred

Name ....

Address ....

City .... State .... Zip ....

☐ BILL ME ☐ Check for $10 enclosed

GIFT — Rip

Name ....

Address ....

City .... State .... Zip ....

☐ BILL ME ☐ Check for $10 enclosed

• BYTE • *Peterborough NH 03458*

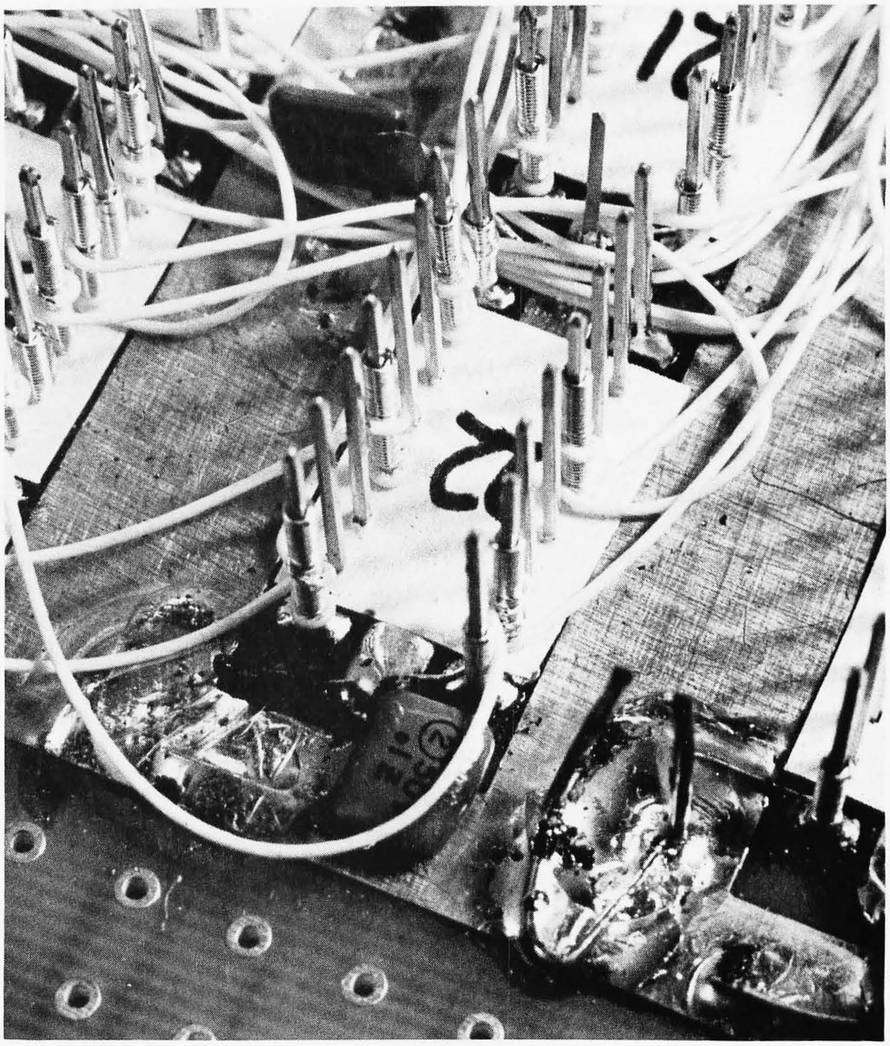


Fig. 3. It is good construction practice with logic circuitry to heavily bypass the logic power supplies. This is especially true of TTL logic, but applies as well to any of the alternative logic families. This photo shows how a typical "decoupling capacitor" of .1 uF is connected between a power bus and the ground plane of the board. The power bus runs between the rows of integrated circuits.

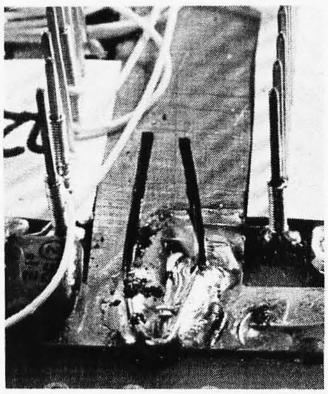


Fig. 4. A technique which is often useful in wiring a neat board is the use of wiring guides. This photo illustrates one very inexpensive way to make such a guide – bend a short piece of wire such as a clipped resistor lead to shape then solder it to the prototyping board's ground plane. In this picture the wiring guide is shown before any wires have been added in its vicinity.

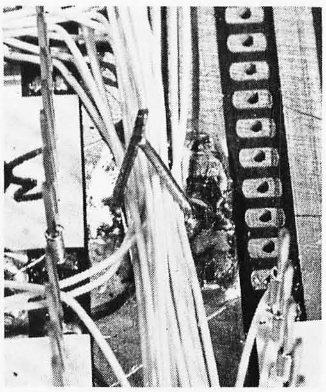
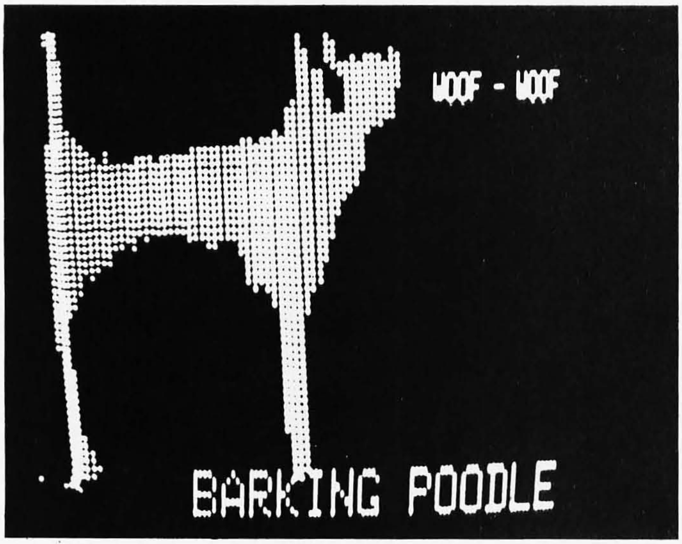


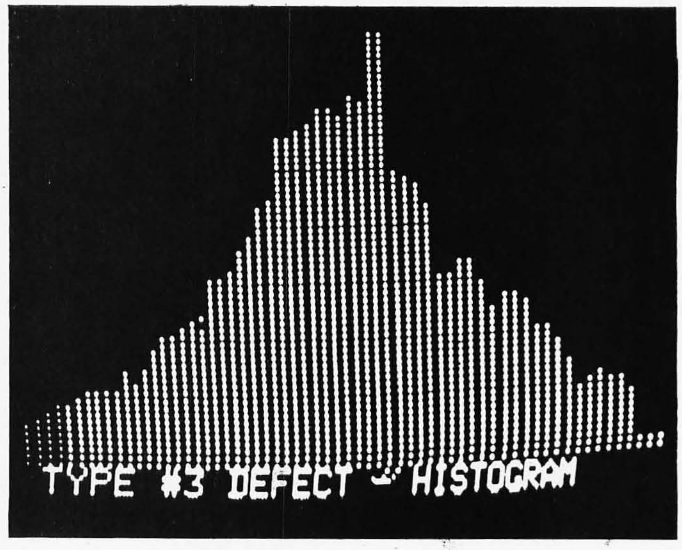
Fig. 5. As wiring progresses, the wiring guides placed at strategic places on the board come into use and become filled with bundles of interconnection wire as in this photograph. After the system is completely debugged, the two ends of the guide can be bent down over the bundle. Until then they are left open at the top so that new wires can be slipped in.

BYTE'S BITS

What You Can Do With an Oscilloscope Graphics Display

Sumner Loomis, Route 1, Box 131, Prairie Point MS 39353, supplies us with some additional illustrations of what can be done with the oscilloscope graphics display unit he built based upon the design of Jim Hogenson in BYTE #2. The barking poodle is self-explanatory. The second picture is a histogram of some experimental data which is being displayed by the graphics output device.





Course in Virginia

A course entitled "Digital Electronics for Automation and Instrumentation" will be given at Virginia Polytechnic Institute and State University, Blacksburg VA, on Dec. 7-12, 1975. Instructors are David G. Larsen, Dr. Peter R. Rony, Jonathan A. Titus and Dr. Frank A. Settle Jr. The course uses *Bugbooks I, II* and *III* as text (see review of *Bugbook III* on page 108). The course is an "in-depth laboratory/lecture course [which] provides hands-on experience with the wiring of digital circuits of modest complexity involving popular and inexpensive TTL integrated circuit chips ..." Enrollment is limited to 24 persons, $325 to members of the American Chemical Society, $360 to non-members. To register or obtain additional information, contact Harold Walsh at the Education Dept., American Chemical Society, 1155 16th St., N.W., Washington DC 20036 (phone 1-202-872-4600). Technical questions about the course can be answered by calling Mr. Larsen at 1-703-951-6478 or Dr. Rony at 1-703-951-6756.

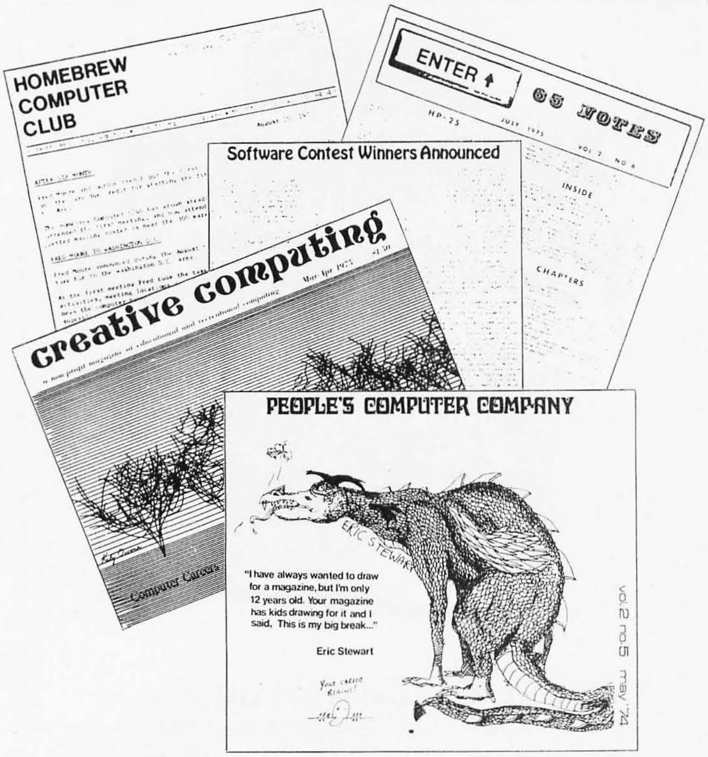
Out of Context . . .

A. M. Biguity provides BYTE with the following note which he found in the Fairchild Semiconductor *34000 Isoplanar CMOS Data Book*, page 2-8:

> "Individuals and tools should be grounded before coming in contact with 34000 devices . . ."

"The trend is to stay upwards compatible with all previous bugs."

Prof. Niklaus Wirth in a 1972 compiler construction course at Stanford University.



Clubs and Newsletters

San Diego Club Activities

The San Diego Computing Society (for lack of a more permanent name) sent BYTE a copy of an August 1975 newsletter. The newsletter included information on club activities as well as a lengthy and wide-ranging comparison of various alternatives for microprocessor users, written by member Dr. Michael Hayes. The article covered possible choices from existing uP evaluation kits to the LSI-11. News of club activities included a scheduled meeting with speakers from Processor Technology Inc. to demonstrate their products which included the new TV-Dazzler color television graphics display device. For information on the San Diego Computing Society, write:

Gary Mitchell
Box 35
Chula Vista CA 92012

Membership dues (which bring the newsletter to your mailbox) are $2.50.

Southern Florida Club?

Roberto R. Denis (11080 N.W. 39 St., Coral Springs FL 33065) inquires about club activity in southern Florida. He's interested in contacting persons and starting a club.

Denver Amateur Computer Society

George Mensik, proprietor of Gateway Electronics in Denver, called BYTE as #4 goes to press. He reports a large and active Denver Amateur Computer Society exists (over 100 members) with interest in computer activities. The DACS meetings include talks by industry people as well as tutorial sessions and special interest gatherings. George's store also serves as a general meeting ground for the Denver area.

You can find out about the DACS by contacting George at Gateway Electronics, 2839 W. 44th Ave., Denver CO 80211 (phone 1-303-458-5444).

News from the Southern California Computer Society

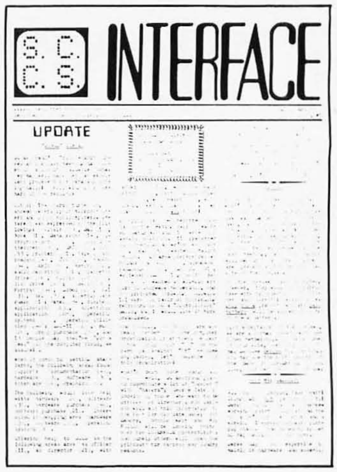
The September 1975 issue of *Interface*, the SCCS's lively newsletter, arrived as BYTE #4 was being assembled in late September. The SCCS people have put together an excellent and informative club letter – running 15 pages in the Vol. 1 No. 1 issue, just for September. In addition to reports on club business matters, the major portion of the letter concerns information and background data of general interest. The newsletter is sent to members. To become a member, send $10 annual dues plus your complete name, address, zip and telephone information to:

Southern California
Computer Society
PO Box 987
South Pasadena CA 91030

In phone conversation with Hal Lashlee of SCCS (1-213-682-3108) BYTE learned a few details of the group purchase of Digital Equipment Corporation LSI-11 computer equipment. The basic deal is that the SCCS will pool individual orders for the DEC equipment to obtain quantity discounts at the 50-, 100- and possibly 200-level price breaks. Orders will be made initially on the assumption of 50-quantity purchases, at the following prices for equipment:

LSI-11 CPU (4K x 16 memory, PDP-11/40 instruction set), $653.
LA-36 DECWriter (30 CPS terminal), $1480.

The SCCS is in the process of securing a fidelity bond for the offer, and, according to Hal, has made arrangements for MasterCharge and BankAmericard payment for orders. A fee for the service – intended as a contribution to the club's activities – is set at a minimum 2% of the order value. Larger contributions



are of course solicited to help pay for the club's operation.

It also looks as if the SCCS will be starting the first active LSI-11 computer users group as a result of this purchase activity.

If you're interested, Hal says that inquiries accompanied by a large self addressed stamped envelope should be sent to the Southern California Computer Society at the address above.

New England Computer Club

An organizational meeting of the New England Computer Club has been scheduled for Nov. 5, 1975. A tentative meeting place, arranged by Ted Poulos of Brookline MA, is at the Jarrell-Ashe plant, Waltham MA. Boston and southern New Hampshire area computer hackers who want to be on the mailing list for future meetings should send their name and address information on an index card or IBM card to:

New England Computer Club
c/o BYTE Magazine
Peterborough NH 03458

Chicago uP Users Group

The Chicago Area Microcomputer Users Group met for the first time in September, with about 30 attending. For latest information, contact Bill Precht, 1102 S. Edson, Lombard IL 60148.

MC 14412 UNIVERSAL MODEM CHIP

MC14412 contains a complete FSK modulator and de-modulator compatible with foreign and USA communications. (0 – 600 BPS)

FEATURES:
- On chip crystal oscillator
- Echo suppressor disable tone generator
- Originate and answer modes
- Simplex, half-duplex, and full duplex operation
- On chip Sine Wave
- Modem self test mode
- Selectable data rates: 0 – 200
  0 – 300
  0 – 600
- Single supply
  VDD = 4.75 to 15 V DC -FL suffix
  VDD = 4.75 to 6 V DC -VL suffix

TYPICAL APPLICATIONS:
- Stand alone – low speed modems
- Built-in low speed modems
- Remote terminals, accoustical couplers

MC 14412 FL 28.99
MC 14412 VL 21.74
6 pages of data .... .60

LED Mounting Hardware
(Specify bezel color)

1750-04 (4 readout) .... $ 7.00
1750-06 (6 readout) .... $ 9.00
1750-08 (8 readout) $11.00
Screen available in red, amber or neutral color

2102-1 MEMORY

1K fully encoded fast (500ns) MOS RAM. New factory prime parts . . . $4.15, 10/$39.00

MC14411 Bit Rate Generator. Single chip for generating selectable frequencies for equipment in data communications such as TTY, printers, CRT's or microprocessors. Generates 14 different standard bit rates which are multiplied under external control to 1X, 8X, 16X or 64X initial value. Built-in crystal oscillator circuit. Operates from single +5V supply.

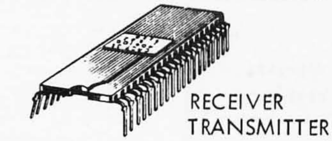
MC14411P with specs $22.40

Output Rates (Hz)

X64	X16	X8	X1
614.4 k	153.6 k	76.8 k	9600
460.8 k	115.2 k	57.6 k	7200
307.2 k	76.8 k	38.4 k	4800
230.4 k	57.6 k	28.8 k	3600
153.6 k	38.4 k	19.2 k	2400
115.2 k	28.8 k	14.4 k	1800
76.8 k	19.2 k	9600	1200
38.4 k	9600	4800	600
19.2 k	4800	2400	300
12.8 k	3200	1600	200
9600	2400	1200	150
8613.2	2153.3	1076.6	134.5
7035.5	1758.8	879.4	109.9
4800	1200	600	75
921.6 k	921.6 k	921.6 k	921.6 k
1.843M	1.843M	1.843M	1.843M

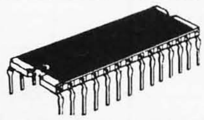
Clock Chip. The MM5318 is an excellent choice for TOD clock all by itself. The display interface has externally selectable character. This allows for outputting one digit at a time to the BCD or 7 seg busses under external control. Useful for computer real time clocks, TV display, serial character transmission. With specs.

MM5318 Clock Chip $7.40



UART-1013A is an ideal device for driving computer peripherals such as teletypes & video terminals. NEW, factory fresh parts. Made by General Instruments. Full specs. New low price!

URT-1013A .... $10.95, 2/$19.95



ASCII KEYBOARD ENCODER

National MM5740AAE. Complete keyboard interface system capable of encoding 90 single pole single throw switch closures into 9-bit ASCII code. Silicon gate technology and is TTL input compatible. Has N Key rollover, one character data storage, repeate function, capacitor key bounce masking, with spec sheet.

MM5740AAEN $19.95

INTERSIL 8038CC

Precision Waveform Generator/Voltage Controlled Oscillator.

Wide Frequency Range of Operation 0:001 Hz to 1.5 MHz

Variable Duty Cycle .... 1% to 99%
#IC8038 . . . with specs $3.95

---

7 SEGMENT DECODE/DRIVER/LATCH



Fairchild 9374 TTL/MSI is a 7 segment decoder driver incorporating input latches and output circuits to directly drive common anode LED displays. Constant current 15 mA sink eliminates need for matching resistors. Automatic zero blanking. Transparent latches. Pin compatible with SN7447.

FDL-9374 .... $2.99

TV CLOCK CHIP SET. Display time of day directly on your TV screen. MM5318 and MM5841 ICs form the basis for video display clock. 5318 keeps time of day (4 or 6 digit) and 5841 provides display video signals with sync derived directly from your set. Makes most interesting way to tell time of day during TV hours.

MM5841/MM5318 Set with specs . . . $20.80



IC SOCKETS. HIGHEST QUALITY!

PC Mount . . . Solder Tail
Skt-0802 . . . 8 pin . . . 25¢, 10/$2.25
Skt-1402 . . . 14 pin . . . 26¢, 10/$2.40
Skt-1602 . . . 16 pin . . . 30¢, 10/$2.70
Skt-1802 . . . 18 pin . . . 45¢, 10/$4.25
Skt-2202 . . . 22 pin . . . 60¢, 10/$5.50
Skt-2402 . . . 24 pin . . . 65¢, 10/$6.00
Skt-4002 . . . 40 pin . . . $1.25, 10/$11.00
Wire Wrap Tails
Skt-1400 . . . 14 pin . . . 49¢, 10/$4.50
Skt-1600 . . . 16 pin . . . 55¢, 10/$5.00

EXAR!

XR-320 Precision timer $1.55
XR-2240 Prog. Counter/timer 4.80
XR-210 FSK Demodulator PLL 5.20
XR-205 Waveform Generator 8.40
XR-2206 Function Generator 5.50
XR-2207 VCO 3.85

EXAR XR-2206KA Function Gen Kit. Includes monolithic function generator IC, PC board, parts list and assembly instruction manual $19.95

PRECISION LAB QUALITY REGULATOR

MC1466L chip can be used to build a high quality voltage and current adjustable power supply. Available output current limited only by the supply and the number of pass transistors. Floating type regulator allows voltage variation from 0 to 250V! .01% line and load regulation.

MC1466L w/specs $8.55

DUAL ± 15V TRACKING REGULATOR

MC1468L 14 pin DIP Package.

FEATURES:
- Output current to 100 mA
- Line and load reg. .06%
- External adj 8 to 20V
- Remote sensing

MC1468L w/specs $4.95

T0-92 VOLTAGE REGULATORS

If your project needs only 100 mA or less, use these tiny plastic regulators right on your PC board. Same size as a plastic transistor. Save space and $. Last 2 digits denote voltage (positive).

LM78L05, LM78L12, LM78L15. 85¢

1/2A Regulator. Fairchild 78MGT2C positive and 79MGT2C negative regulators are adjustable from 5V to 30V with only 2 external resistors. Only 1 resistor for +5V (±.2V) from the 78 MGT2C. Comes in 8 pin minidip package format with heat tabs for attaching to PC board or heat sink. Easily connected for ± tracking supplies.

78MGT2C (pos) or 79MGT2 (neg) $2

5V, 3A T0-3 Regulator. Fairchild SH323KC monolithic regulator in T0-3 power package. Use it with the same ease of installation as the famous 309K (same pin arrangement). Take care of those heavy current requirements without separate regulator/pass transistor combinations.

SH323K3 5V, 3A $9.95

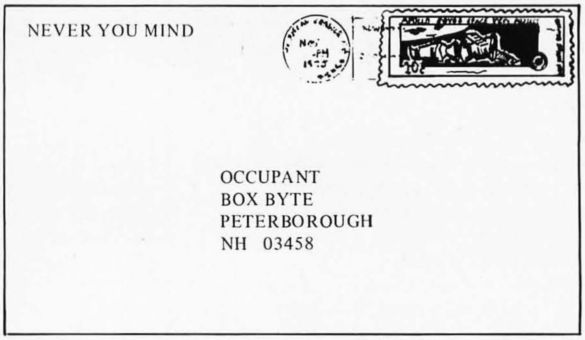
MINIATURE ROCKER DIP SWITCHES

Dual in-line SPST switch arrays for P.C. mounting. Spring loaded sliding ball contact system for positive, tease proof contact. Comes in contact arrangements from 4 to 10 per pack. Fits standard DIP sockets. Last two digits of stock # indicate number of switches.

DIS-76B04 .... $3.10
DIS-76B06 .... $3.50
DIS-76B07 .... $3.75
DIS-76B08 .... $3.95
DIS-76B10 .... $4.35

---

tri-tek, inc. P.O. BOX 14206, DEPT 7 PHOENIX, ARIZONA 85063



BIT COLLECTING

Editor,

First let me congratulate you and your staff on an extremely interesting and informative first issue. It is definitely the best first issue of a magazine that I have ever read. I, like thousands of others, decided on subscribing to BYTE after collecting a few bits of information concerning what you intended to publish. I was very pleased to see that you managed to get all the bits together to form a BYTE! I believe that while the MITS people have set the standard for "the affordable computer," the BYTE people are now setting the standard of how to educate people in using it!

I was particularly interested in the comments of one reader regarding the playing of games between computers. I intend to develop a CHESS program for my Altair 8800, probably in MITS 8k BASIC language. I would like to correspond with all the people out there who share the same interest. While on this topic, I should say that I would like to see some competitions between small computers classified according to core size, instruction set, cost or whatever else will encourage people to pack the most efficiency and genius into the lowest cost system.

In closing, I must make one final observation. I noticed in Don Lancaster's article on "Serial Interface," p. 35 [BYTE #1], that the originate modem tx and answer modem rx frequencies do not agree as is necessary for proper operation. The answer modem receive frequencies should be 1070 Hz space and 1270 Hz mark. Incidentally, I thought the article by Don was of his usual high calibre and should serve as an example for others to follow.

Dan Clarke
105 Fir Court
Fredericton NB
Canada E3A 2E9

DOWN ON EARTH?

Editor, BYTE:

I ordered my subscription to BYTE when the first ad was printed in *73 Magazine* last summer. The first book I got was #2 (October 1975). Where is my #1? Please send it.

I have a good understanding of analog and logical circuitry – i.e., gates, counters, shift registers, radio, TV and machine control. But –

The only down on earth articles from which I got the full meaning in issue 2 were "Television Interface" (best by far), "Quick Test of Keyboards," and "Asynchronitis."

Please – I want to understand 8008 and 8080. What's inside, relating to the logic I know? When are they called for? And what must be used with them?

Most articles start in the process of program development – much too late for the beginner. It seems many *cute* words add to the snow: Like byte, Basic, life, architecture, kluge??, algorithm, subroutines, language – various,

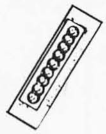
5% OFF ON ORDERS OVER $50.00
10% OFF ON ORDERS OVER $100.00
15% OFF ON ORDERS OVER $250.00

DECEMBER SPECIALS

TTL

7442	$ .59
7447	.69
7489	1.99
74153	.69
74193	.99

9 DIGIT LED DISPLAY FNA 37



On multiplexed substrate, comm. cathode compatable with all 8 digit calculator chips, 7 segment right hand decimal, red with clear magnifying lens, .12" character, 1 to 4 MA, 1.8 V typ 2⅜" x ¼" x ³⁄₁₆" high **$3.95**

LINEAR

324	$1.19
380-8	1.09
565	1.49
739	.89
75491	.59

CLOCK CIRCUITS

5311	$3.95
5312	2.95
5314	3.95

MEMORIES

1702A	$14.95
2101	2.95
2102-1	3.25
2102-2	3.25
5203	12.95
5260	.99
5261	.99
5262	2.95

NINE DIGIT SPERRY GAS DISCHARGE DISPLAY

SP-425-09 1.25" x 3" overall — .25" digits — connects to 18 led edge connector — hi voltage — prime quality **$1.49**

DVM CHIP 4½ DIGIT

MM5330 — P channel device provides all logic for 4½ digit digital volt meter 16 pin DIP with data **$14.95**

CALCULATOR CHIPS

5001	$1.49
5002	1.98
5005	2.49

TTL

		7451	.17	74154	1.25
7400	$ .14	7453	.17	74155	1.07
7401	.16	7454	.17	74156	1.07
7402	.15	7460	.17	74157	.99
7403	.16	7464	.35	74158	1.79
7404	.19	7465	.35	74160	1.39
7405	.19	7470	.30	74161	1.25
7406	.35	7472	.30	74162	1.49
7407	.35	7473	.35	74163	1.39
7408	.18	7474	.35	74164	1.59
7409	.19	7475	.57	74165	1.59
7410	.16	7476	.39	74166	1.49
7411	.25	7483	.79	74170	2.30
7413	.55	7485	1.10	74173	1.49
7416	.35	7486	.40	74174	1.62
7417	.35	7489	2.48	74175	1.39
7420	.16	7490	.59	74176	.89
7422	.26	7491	.97	74177	.84
7423	.29	7492	.71	74180	.90
7425	.27	7493	.60	74181	2.98
7426	.26	7494	.94	74182	.79
7427	.29	7495	.79	74184	2.29
7430	.20	7496	.79	74185	2.29
7432	.23	74100	1.30	74187	5.95
7437	.35	74105	.44	74190	1.35
7438	.35	74107	.40	74191	1.35
7440	.17	74121	.42	74192	1.25
7441	.98	74122	.45	74193	1.19
7442	.77	74123	.85	74194	1.25
7443	.87	74125	.54	74195	.89
7444	.87	74126	.63	74196	1.25
7445	.89	74141	1.04	74197	.89
7446	.93	74145	1.04	74198	1.79
7447	.89	74150	.97	74199	1.79
7448	1.04	74151	.79	74200	5.90
7450	.17	74153	.99		

LOW POWER TTL

74L00	$ .25	74L51	$ .29	75L90	$1.49
74L02	.25	74L55	.33	74L91	1.45
74L03	.25	74L71	.25	74L93	1.69
74L04	.25	74L72	.39	74L95	1.69
74L06	.25	74L73	.49	74L98	2.79
74L10	.25	74L74	.49	74L164	2.79
74L20	.33	74L78	.79	74L165	2.79
74L30	.33	74L85	1.25		
74L42	1.49	74L86	.69		

HIGH SPEED TTL

74H00	$ .25	74H21	$ .25	74H55	$ .25
74H01	.25	74H22	.25	74H60	.25
74H04	.25	74H30	.25	74H61	.25
74H08	.25	74H40	.25	74H62	.25
74H10	.25	74H50	.25	74H72	.39
74H11	.25	74H52	.25	74H74	.39
74H20	.25	74H53	.25	74H76	.49

8000 SERIES

8091	$ .53	8214	$1.49	8811	$ .59
8092	.53	8220	1.49	8812	.89
8095	1.25	8230	2.19	8822	2.19
8121	.80	8520	1.16	8830	2.19
8123	1.43	8551	1.39	8831	2.19
8130	1.97	8552	2.19	8836	.25
8200	2.33	8554	2.19	8880	1.19
8210	2.79	8810	.69	8263	5.79
				8267	2.59

9000 SERIES

9002	$ .35	9309	$ .79	9601	$ .89
9301	1.03	9312	.79	9602	.79

CMOS

		4016A	.56	4050A	.59
4000A	$ .26	4017A	1.19	4066A	.89
4001A	.25	4020A	1.49	4068A	.44
4002A	.25	4021A	1.39	4069A	.44
4006A	1.35	4022A	1.10	4071A	.26
4007A	.26	4023A	.25	4072A	.35
4008A	1.79	4024A	.89	4073A	.39
4009A	.57	4025A	.25	4075A	.39
4010A	.54	4027A	.59	4078A	.39
4011A	.29	4028A	.98	4081A	.26
4012A	.25	4030A	.44	4082A	.35
4013A	.45	4035A	1.27	4528A	1.60
4014A	1.49	4042A	1.47	4585A	2.10
4015A	1.49	4049A	.59		

74C00	$ .22	74C74	$1.04	74C162	$2.93
74C02	.26	74C76	1.34	74C163	2.66
74C04	.44	74C107	1.13	74C164	2.66
74C08	.68	74C151	2.61	74C173	2.61
74C10	.35	74C154	3.15	74C195	2.66
74C20	.35	74C157	1.76	80C95	1.35
74C42	1.61	74C160	2.48	80C97	1.13
74C73	1.04	74C161	2.93		

NEW ITEM — EXCLUSIVE WITH IEU

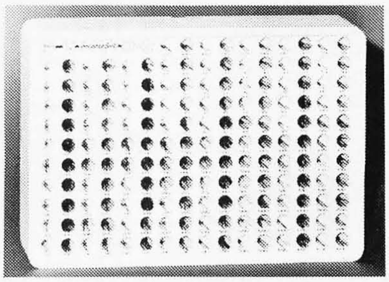
RESISTOR ASSORTMENTS

SUPPLIED WITH CONVENIENT STORAGE UNIT
DESIGNED FOR EXPERIMENTERS, TECHNICIANS AND SCHOOLS

RS 1-25 $99.50

AN ASSORTMENT OF
2725 RESISTORS
170 VALUES FROM
.51 OHMS TO 5.6M OHMS
QUANTITIES DETERMINED
BY TYPICAL USAGE
¼ WATT 5%
CARBON FILM

+ $1.00 SHIPPING
& HANDLING
SHIPPING VIA UPS OR
PARCEL POST



REPLACEMENTS IMMEDIATELY
AVAILABLE AT LOW COST
BY MAIL FROM IEU

Rohm
QUALITY · RELIABILITY

RS 2-25 $114.50

AN ASSORTMENT OF
4000 RESISTORS
100 EACH OF 40
STANDARD VALUES
1 OHM TO 3.3M OHM
¼ WATT 5%
CARBON FILM

RS 3-50 $69.50

AN ASSORTMENT OF
2000 RESISTORS
50 EACH OF 40
STANDARD VALUES
1 OHM TO 3.3M OHMS
½ WATT 5%
CARBON FILM

Data sheets on request. With order add $.30 for items less than $1.00 ea.

LED's

MV10B	Red TO 18	$ .22
MV50	Axial leads	.18
MV5020	Jumbo Vis. Red (Red Dome)	.22
	Jumbo Vis. Red (Clear Dome)	.22
ME4	Infra red diff. dome	.54
MAN1	Red 7 seg. .270"	2.19
MAN2	Red alpha num .32"	4.39
MAN4	Red 7 seg. .190"	1.95
MAN5	Green 7 seg. .270"	3.45
MAN6	.6" high solid seg.	4.25
MAN7	Red 7 seg. .270"	1.19
MAN3	Red 7 seg. .127" straight pins	.29
MAN8	Yellow 7 seg. .270"	3.45
MAN66	.6" high spaced seg.	3.75
MCT2	Opto-iso transistor	.61

MULTIPLE DISPLAYS

NSN33	3 digit .12" red led 12 pin fits IC skt.	$1.79
HP45082 7405	5 digit .11 led magn. lens com. cath	3.49
HP5082 7414	4 digit .11 LED magn. lens comm. cath.	3.25
FNA37	9 digit 7 seg led RH dec clr. magn. lens	4.95
SP-425-09	9 digit .25" neon direct interface with MOS/LSI, 180 VDC, 7 seg.	1.79

SHIFT REGISTERS

MM5013	1024 bit accum. dynamic mDIP	$1.75
MM5016	500/512 bit dynamic mDIP	1.59
SL5-4025	QUAD 25 bit	1.29

DTL

930	$ .15	937	.15	949	.15
932	.15	944	.15	962	.15
936	.15	946	.15	963	.15

MEMORIES

1101	256 bit RAM MOS	$ 1.50
1103	1024 bit RAM MOS	3.95
1702A	2048 bit static PROM UV eras.	17.95
2102-2	1024 bit static RAM	4.25
5203	2048 bit UV eras PROM	17.95
5260	1024 bit RAM	2.49
5261	1024 bit RAM	2.69
5262	2048 bit RAM	5.95
7489	64 bit ROM TTL	2.48
8223	Programmable ROM	3.69
74200	256 bit RAM tri-state	5.90

CALCULATOR & CLOCK CHIPS

5001	12 DIG 4 funct fix dec	$2.49
5002	Same as 5001 exc btry power	2.79
5005	12 DIG 4 fuct w/mem	2.99
MM5725	8 DIG 4 funct chain & dec	1.98
MM5736	18 pin 6 DIG 4 funct	4.45
MM5738	8 DIG 5 funct K & mem	5.35
MM5739	9 DIG 4 funct (btry sur)	5.35
MM5311	28 pin BCD 6 dig mux	4.45
MM5312	24 pin 1 pps BCD 4 dig mux	3.95
MM5313	28 pin 1 pps BCD 6 dig mux	4.45
MM5314	24 pin 6 dig mux	4.45
MM5316	40 pin alarm 4 dig	5.39

LINEAR CIRCUITS

300	Pos V Reg (super 723) TO-5	$ .71
301	Hi Perf Op Amp mDIP TO-5	.29
302	Volt follower TO-5	.53
304	Neg V Reg TO-5	.80
305	Pos V Reg TO-5	.71
307	Op AMP (super 741) mDIP TO-5	.26
308	Mciro Pwr Op Amp mDIP TO-5	.89
309K	5V 1A regulator TO-3	1.35
310	V Follower Op Amp mDIP	1.07
311	Hi perf V Comp mDIP TO-5	.95
319	Hi Speed Dual Comp DIP	1.13
320	Neg Reg 5.2, 12, 15 TO-3	1.19
322	Precision Timer DIP	1.70
324	Quad Op Amp DIP	1.52
339	Quad Comparator DIP	1.58
340K	Pos V reg (5V, 6V, 8V, 12V, 15V, 18V, 24V) TO-3	1.69
340T	Pos V reg (5V, 6V, 8V, 12V, 15V, 18V, 24V) TO-220	1.49
372	AF-IF Strip detector DIP	2.93
373	AM/FM/SSB Strip DIP	.53
376	Pos V Reg mDIP	2.42
377	2w Stereo amp DIP	1.16
380	2w Audio Amp DIP	1.13
380-8	.6w Audio Amp mDIP	1.52
381	Lo Noise Dual preamp DIP	1.52
382	Lo Noise Dual preamp DIP	.71
550	Prec V Reg DIP	.89
555	Timer mDIP	.89
556A	Dual 555 Timer DIP	1.49
560	Phase Locked Loop DIP	2.48
562	Phase Locked Loop DIP	2.48
565	Phase Locked Loop DIP TO-5	2.38
566	Function Gen mDIP TO-5	2.25
567	Tone Decoder mDIP	2.66
709	Operational AMP TO-5 or DIP	.26
710	Hi Speed Volt Comp DIP	.35
711	Dual Difference Compar DIP	.26
723	V Reg DIP	.62
739	Dual Hi Perf Op Amp DIP	1.07
741	Comp Op Amp mDIP TO-5	.32
747	741 Dual Op Amp DIP or TO-5	.71
748	Freq Adj 741 mDIP	.35
1304	FM Mulpx Stereo Demod DIP	1.07
1307	FM Mulpx Stereo Demod DIP	.74
1458	Dual Comp Op Amp mDIP	.62
1800	Stereo multiplexer DIP	2.48
LH2111	Dual LM 211 V Comp DIP	1.70
3900	Quad Amplifier DIP	.35
7524	Core Mem Sense AMPL DIP	.71
8038	Voltage contr. osc. DIP	4.25
8864	9 DIG Led Cath Drvr DIP	2.25
75150	Dual Line Driver DIP	1.75
75451	Dual Perepheral Driver mDIP	.35
75452	Dual Peripheral Driver mDIP	.35
75453	(351) Dual Periph Driver mDIP	.35
75491	Quad Seq Driver for LED DIP	.71
75492	Hex Digit driver DIP	.80

Satisfaction guaranteed. Shipment will be made via first class mail in U.S., Canada and Mexico within 3 days from receipt of order. Add $.50 to cover shipping and handling for orders under $25.00. Minimum order $5.00. California residents add sales tax.

INTERNATIONAL ELECTRONICS UNLIMITED
P.O. BOX 1708/ MONTEREY, CA. 93940 USA
PHONE (408) 659-3171

compilers, stack, assembler, mnemonic, opcodes, disavowal.

A uP could be used as the heart of an automatic bowling scorer, getting pinfall data, machine position data, bowler ID data and special information for error correction – and in turn would control the printer (or TV display) error correction display, etc. I have special interest in such a machine! A machine exists, implemented with IC logic – but what power has a uP?? Programming the uP must come after a good knowledge of its inner workings and a good *mental model* is developed.

I hope BYTE magazine will try to get to the basics of computing using the uP. What jobs are they fitted to? Could an 8080 be used for:

1. A complex pin-ball machine control?
2. A computer like HP-65?
3. Bowling computer?
4. Automatic machine controls – various.

I hope your magazine will direct some effort towards people with needs and interests like mine.

Pat Murphy
Muskegon MI

You should be straightened out on the circulation error by now, Pat. We've had some articles on basics to date, and will continue to do so in future issues.

LOGICAL COMPLEMENT

Dear Mr. Helmers,

I was struck by your editorial in the October 1975 issue, "The Home Brew Voder," because I have long wanted to be able to use a small computer for exactly the opposite purpose. I want to be able to speak into a microphone and have an electric typewriter type out the words I am saying.

I am a writer and, speaking as a writer, it has always been the most tedious (if necessary) chore when researching material to have to copy out my source material by hand, to make notes by hand, and then to rewrite my first drafts by hand (and by "hand" I mean by typewriter also). It would be much easier and more accurate when researching source material especially to be able to read the material out loud and have an accurate typescript made automatically, or, what amounts to the same thing, to read the material into a tape recorder for later transcription . . .

. . . If you or any of your readers can convert this idea of mine into practical circuits, I would certainly appreciate it, and I am sure the idea is commercially viable as well.

M. Krieger
37 Eighth Ave.
Brooklyn NY 11217

THE AUDIO CASSETTE TOWER OF BABEL

Dear Mr. Helmers,

Congratulations! I am glad to see a real magazine exclusively devoted to the "do it yourself" computer enthusiast – now I don't have to buy an audio or electronic magazine for one article about computers.

You are right about the three areas to cover – hardware, software and applications. However, there is another area that ties these three together and without it we would have chaos – that area is standards!

Now that so many manufacturers are getting into tne computer field everyone wants to do their own thing. That's fine – but, we do need standards. It is not really important what goes on in the "black box" or how fancy the operating system is as long as we can swap programs and they will run without having to rewrite them.

As the first magazine in the field, you can be the driving force in helping to prevent a "Tower of Babel." We need a specification spread sheet so that

7400N TTL

SN7400N	.16	SN7451N	.27	SN74151N	1.25
SN7401N	.16	SN7453N	.27	SN74153N	1.35
SN7402N	.21	SN7454N	.41	SN74154N	1.25
SN7403N	.16	SN7459A	.25	SN74155N	1.21
SN7404N	.21	SN7460N	.22	SN74156N	1.30
SN7405N	.24	SN7470N	.45	SN74157N	1.30
SN7406N	.45	SN7472N	.39	SN74160N	1.75
SN7407N	.45	SN7473N	.45	SN74161N	1.45
SN7408N	.25	SN7474N	.45	SN74163N	1.65
SN7409N	.25	SN7475N	.80	SN74164N	1.65
SN7410N	.20	SN7476N	.47	SN74165N	1.65
SN7411N	.30	SN7480N	.50	SN74166N	1.70
SN7412N	.42	SN7482N	1.75	SN74167N	5.50
SN7413N	.85	SN7483N	1.15	SN74170N	3.00
SN7414N	.70	SN7485N	1.12	SN74172N	18.00
SN7416N	.43	SN7486N	.45	SN74173N	1.70
SN7417N	.43	SN7488N	3.50	SN74174N	1.95
SN7418N	.25	SN7489N	3.00	SN74175N	1.95
SN7420N	.21	SN7490N	.59	SN74176N	.90
SN7421N	.39	SN7491N	1.20	SN74177N	.90
SN7423N	.37	SN7492N	.82	SN74180N	1.05
SN7425N	.43	SN7493N	.82	SN74181N	3.55
SN7426N	.31	SN7494N	.91	SN74182N	.95
SN7427N	.37	SN7495N	.91	SN74184N	2.30
SN7429N	.42	SN7496N	.91	SN74185N	2.20
SN7430N	.26	SN74100N	1.25	SN74187N	6.00
SN7432N	.31	SN74107N	.49	SN74190N	1.50
SN7437N	.47	SN74121N	.55	SN74191N	1.50
SN7438N	.40	SN74122N	.49	SN74192N	1.50
SN7439A	.25	SN74123N	1.05	SN74193N	1.40
SN7440N	.21	SN74125N	.60	SN74194N	1.45
SN7441N	1.10	SN74126N	.81	SN74195N	1.00
SN7442N	1.08	SN74132N	3.00	SN74196N	1.25
SN7443N	1.05	SN74141N	1.15	SN74197N	1.00
SN7444N	1.10	SN74142N	6.50	SN74198N	2.25
SN7445N	1.10	SN74143N	7.00	SN74199N	2.25
SN7446N	1.15	SN74144N	7.00	SN74200N	7.00
SN7447N	.89	SN74145N	1.15	SN74251N	2.50
SN7448N	.99	SN74148N	2.50	SN74284N	6.00
SN7450N	.26	SN74150N	1.10	SN74285N	6.00

20% Discount for 100 Combined 7400's

CMOS

CD4000	.25			74C10N	.65
CD4001	.25			74C20N	.65
CD4002	.25	CD4030	.65	74C30N	.65
CD4006	2.50	CD4035	1.85	74C42N	2.15
CD4007	.25	CD4040	2.45	74C73N	1.50
CD4009	.59	CD4042	1.90	74C74	1.15
CD4010	.59	CD4044	1.50	74C90N	3.00
CD4011	.25	CD4046	2.51	74C95N	2.00
CD4012	.25	CD4047	2.75	74C107N	1.25
CD4013	.47	CD4049	.79	74C151	2.90
CD4016	.56	CD4050	.79	74C154	3.00
CD4017	1.35	CD4051	2.98	74C157	2.15
CD4019	.55	CD4053	2.98	74C160	3.25
CD4020	1.49	CD4060	3.25	74C161	3.25
CD4022	1.25	CD4066	1.75	74C163	3.00
CD4023	.25	CD4069	.45	74C164	3.25
CD4024	1.50	CD4071	.45	74C173	2.60
CD4025	.25	CD4081	.45	74C193	2.75
CD4027	.69	74C00N	.39	74C195	2.75
CD4028	1.65	74C02N	.55	80C97	1.50
CD4029	2.90	74C04N	.75		

LINEAR

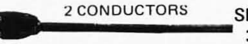
LM100H	15.00			LM1310N	2.95
LM106H	2.50			LM1351N	1.65
LM171H	3.75	LM373N	3.25	LM1414N	1.75
LM212H	7.00	LM377N	4.00	LM1458C	.65
LM300H	.80	LM380N	1.39	LM1496N	.95
LM301H	3/1.00	LM380CN	1.05	LM1556V	1.85
LM301CN	3/1.00	LM381N	1.79	LM2111N	1.95
LM302H	.75	LM382N	1.79	LM2901N	2.95
LM304H	1.00	NE501K	8.00	LM3065N	.69
LM305H	.95	NE510A	6.00	LM3900N	.55
LM307CN	.35	NE531H	3.00	LM3905N	.60
LM308H	1.00	NE536T	6.00	LM5556N	1.85
LM308CN	1.00	NE540L	6.00	MC5558V	1.00
LM309H	1.10	NE550N	.79	LM7525N	.90
LM309K	1.25	NE553	2.50	LM7528N	2.20
LM310CN	1.15	NE555V	.49	LM7534N	2.20
LM311H	.90	NE565H	.99	LM7535N	1.25
LM311N	.90	NE565N	1.25	8038B	4.95
LM318CN	1.50	NE566CN	1.95	LM75450	.49
LM319N	1.30	NE567H	1.25	75451CN	.39
LM319D	9.00	NE567V	1.50	75452CN	.39
LM320K 5	1.35	LM703CN	.45	75453CN	.39
LM320K 5.2	1.35	LM709H	.29	75454CN	.39
LM320K 12	1.35	LM709N	.29	75491CN	.79
LM320K 15	1.35	LM710N	.79	75492CN	.89
LM323K-5	10.50	LM711N	.39	75494CN	.89
LM324N	1.80	LM723N	.55	**RCA LINEAR**	
LM339N	1.70	LM723H	.55	CA3013	1.70
LM340K 5	1.95	LM733N	1.00	CA3023	2.15
LM340K 12	1.95	LM739N	1.29	CA3035	2.25
LM340K 15	1.95	LM741CH	3/1.00	CA3039	1.35
LM340K 24	1.95	LM741CN	3/1.00	CA3046	1.15
LM340To 5	1.75	LM741 14N	.39	CA3059	2.46
LM340To 6	1.75	LM747H	.79	CA3060	2.80
LM340To 12	1.75	LM747N	.79	CA3080	.85
LM340To 15	1.75	LM748H	.39	CA3083	1.60
LM340To 24	1.75	LM748N	.39	CA3086	.59
LM350N	1.00	LM1303N	.90	CA3089	3.25
LM351CN	.65	LM1304N	1.19	CA3091	8.25
LM370N	1.15	LM1305N	1.40	CA3123	1.85
LM370H	1.15	LM1307N	.85	CA3600	1.75

DATA HANDBOOKS

7400	Pin-out & Description of 5400/7400 ICS	$2.95
CMOS	Pin-out & Description of 4000 Series ICS	$2.95
LINEAR	Pin-out & Functional Description of Linear Circuits	$2.95

6' POWER SUPPLY CORDS ....

2 CONDUCTORS



SPECIAL 3/$1.00

THUMBWHEEL SWITCHES

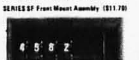


THUMBWHEEL SWITCH ONLY
SF – Front Mount   SR – Rear Mount

Part No.	Description	Price
SF-12	Single Pole 10 Position	$2.50
SR-12	Decimal	3.00
SF-21	10 Position BCD only	2.50
SR-21		3.00

Snap together – No Hardware

SERIES SF Front Mount Assembly ($11.70)



Note: Picture to the left has the following 4 ea. SF-12, 1 ea. SF-EP, 1 ea. SF-BB, 1 ea. SF-DP and 1 ea. SF-HB.

ACCESSORIES

Part No.	Description	Price
SF-EP	End Plate (Pair)	.50
SF-DP	Divider Plate (each)	.40
SF-BB	Blank Body (each)	.40
SF-HB	Half Body (each)	.40

SERIES SR Rear Mount Assembly ($13.70)



Note: Picture to the left has the following 4 ea. SR-12, 1 ea. SR-EP, 1 ea. SR-BB, 1 ea. SR-DP and 1 ea. SR-HB

ACCESSORIES

Part No.	Description	Price
SR-EP	End Plates (pair)	$ .50
SR-DP	Divider Plate (each)	.40
SR-BB	Blank Body (each)	.40
SR-HB	Half Body (each)	.40

Ordering: Order desired switch or switches and add necessary accessories for your particular application.

JAMES CHRISTMAS SPECIALS

DIODES

IN4001	50V @ 1 Amp	15/$1.00
IN4002	100V @ 1 Amp	15/$1.00
IN4003	200V @ 1 Amp	15/$1.00
IN4004	400V @ 1 Amp	15/$1.00
IN4148	Switching	20/$1.00

TRANSISTORS

2N2907A	PNP Switching	6/$1.00
2N2222A	NPN Switching	6/$1.00
2N3904	NPN Amp	6/$1.00
2N3906	PNP Amp	6/$1.00
2N918	NPN RF	6/$1.00
2N5951	NJ Fet	6/$1.00
C106B1	3.6 Amp SCR	2/$1.00

TTL/LINEAR

7400	Gate	7/$1.00
7447	Decoder	.79
7490	Counter	.49
74100	8 Bit Latch	.99
74154	Decoder	.99
74197	Counter	.75
LM301H–LM741H		4/$1.00
LM324N–Quad 741		1.19
LM309H	5V Reg TO-5	.79
LM309K	5V Reg TO-3	.99

MM5262 DYNAMIC 2K RAM $2.95

CLOCK & CALC. CHIPS

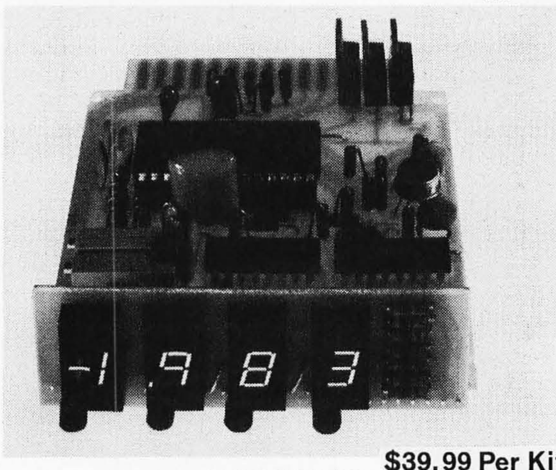
MM5311N	6 Digit	$2.95
MM5312N	4-Digit	2.95
MM5313N	6 Digit	2.95
MM5314N	6 Digit	2.95
MM5316N	6 Digit Alarm	3.95
MM5725N	8 Digit-4 Funt.	1.85
MM5736N	6 Digit-4 Funt.	1.95
MM5738N	8 Digit-5 Funt.	2.95

26 Awg RIBBON CABLE

1 Ft. Minimum	1-9 ft.	10 ft.
4 Cond.	.29 ft.	.25 ft.
8 Cond.	.59 ft.	.49 ft.
16 Cond.	.89 ft.	.79 ft.
32 Cond.	1.89 ft.	1.69 ft.

DIGITAL-VOLTMETER

JE801 DVM



GENERAL DESCRIPTION

$39.99 Per Kit
SINGLE UNIT

The JE801 is a three and one half digit, auto polarity digital voltmeter, in a kit form. It features several options not available in any commercial digital voltmeter. Its low cost is perhaps the most important feature, which is achieved by offering it in a kit form. A kit allows the unit to be used by small OEM's where cost effectiveness is an important factor, and by the hobbyist who has to be concerned with cost. The unit also features on card regulators, allowing it to be operated off a single plus and minus fifteen volt, unregulated power supply. The unit has a small size of three inches width, three and three quarters of an inch length, and one and a quarter inch height.

DISPLAY LEDS

FND70	Com. Cath.	.250	.59
MAN 1	Com. Ano.	.270	$1.95
MAN 2	5x7 Matrix	.300	3.95
MAN 3	Com. Cath.	.125	.39
MAN 4	Com. Cath.	.187	1.95
MAN 7	Com. Ano.	.30	1.50
DL33	Com. Cath.	.125	1.95
DL747	Com. Ano.	.625	1.95



DISCRETE LEDS

MV 10	Red	5/$1.00
MV 50	Red	6/$1.00
MV 5024	Red	5/$1.00
MV 5024	Green	4/$1.00
MV 5024	Yellow	4/$1.00
MV 5024	Orange	4/$1.00
MV 50*	Red 1¼" Leads	5/$1.00



IC SOLDERTAIL – LOW PROFILE (TIN) SOCKETS

	1-24	25-49	50-100		1-24	25-49	50-100
8 pin	$.17	.16	.15	24 pin	$ .38	.37	.36
14 pin	.20	.19	.18	28 pin	.45	.44	.43
16 pin	.22	.21	.20	36 pin	.60	.59	.58
18 pin	.29	.28	.27	40 pin	.63	.62	.61
22 pin	.36	.37	.36				

SOLDERTAIL STANDARD (GOLD)

8 pin	$.30	.27	.24	24 pin	$ .70	.63	.57
14 pin	.35	.32	.29	28 pin	1.10	1.00	.90
16 pin	.38	.35	.32	36 pin	1.55	1.40	1.26
18 pin	.52	.47	.43	40 pin	1.75	1.59	1.45

WIRE WRAP SOCKETS (GOLD) LEVEL #3

10 pin	$.45	.41	.37	24 pin	$1.05	.95	.85
14 pin	.39	.38	.37	28 pin	1.40	1.25	1.10
16 pin	.43	.42	.41	36 pin	1.59	1.45	1.30
18 pin	.75	.68	.62	40 pin	1.75	1.55	1.40

50 PCS. RESISTOR ASSORTMENTS $1.75 per assort.

Each assortment contains 50 pcs. of ¼ watt, 5% resistors

ASST. 1 5 ea:	10 OHM	12 OHM	15 OHM	18 OHM	22 OHM
	27 OHM	33 OHM	39 OHM	47 OHM	56 OHM
ASST. 2 5 ea:	68 OHM	82 OHM	100 OHM	120 OHM	150 OHM
	180 OHM	220 OHM	270 OHM	330 OHM	390 OHM
ASST. 3 5 ea:	470 OHM	560 OHM	680 OHM	820 OHM	1K
	1.2K	1.5K	1.8K	2.2K	2.7K
ASST. 4 5 ea:	3.3K	3.9K	4.7K	5.6K	6.8K
	8.2K	10K	12K	15K	18K
ASST. 5 5 ea:	22K	27K	33K	39K	47K
	56K	68K	82K	100K	120K
ASST. 6 5 ea:	150K	180K	220K	270K	330K
	390K	470K	560K	680K	820K
ASST. 7 5 ea.	1M	1.2M	1.5M	1.8M	2.2M
	2.7M	3.3M	3.9M	4.7M	5.6M

14 PCS. POTENTIOMETER ASSORTMENTS $9.95

Each assortment contains 14 pcs of 10 turn pots.

ASST. A 2 ea: 10 OHM-20 OHM-50 OHM-160 OHM 200 OHM-250 OHM-500 OHM
ASST. B 2 ea: 1K, 2K, 2.5K, 10K, 20K, 25K, 50K
ASST. C 2 ea: 50K, 100K, 200K, 250K, 500K, 1M, 2M

All pots are available in single unit quantities. **$.99 ea.**

DPST C & K ROCKER SWITCH

They are rated at 125 Vac @ 5A. They are excellent in application such as Microcomputer Panel Switches **$0.69**

Dim: 1"x1"x¼"

PRIME INTEGRATED CIRCUIT ASSORTMENTS

ASST. 8	3 ea.	SN7400 SN7410	7401 7430	7402 7438	7403 7440	7404 7472	SSI/TTL	$5.95 ASST.
ASST. 9	2 ea.	SN7447 SN74145	7490 74175	7491 74180	74100 74191	74121 74193	MSI/TTL	$9.95 ASST.
ASST. 10	2 ea.	CD4001 CD4016	4002 4017	4011 4019	4012 4023	4013 4030	CMOS	$7.95 ASST.
ASST. 11	2 ea.	LM301T LM311T	301N 565T	302T 567T	307T 741T	309K 741N	LINEAR	$10.95 ASST.

Satisfaction Guaranteed. $5.00 Min. Order. U.S. Funds.
California Residents — Add 6% Sales Tax
Write for FREE 1976 Catalog — Data Sheets .25¢ each

JAMES

P.O. BOX 822, BELMONT, CA. 94002
PHONE ORDERS — (415) 592-8097

MICROPROCESSOR COMPONENTS

8008 – **$19.95**   2102 – **$2.95**

	CPU'S			RAM'S		
8008	8 bit CPU	$19.95	1101	256x1	Static	$ 2.25
8080	8 bit Super 8008	149.95	1103	1Kx1	Dynamic	2.95
4040	4 bit 8080	39.95	2102	1Kx1	Static	2.95
	UARTS		2107	4Kx1	Dynamic	19.95
AY – 5 - 1013		6.95	7010	1Kx1	Mnmos	29.95
	PROM'S		7489	16x4	Static	2.49
8223	32x8 Prom	3.00	8101	1Kx1	Static	7.95
1702A	2K E Prom	15.95	8111	1Kx1	Static	7.95
5203Q	2K E Prom	14.95	8599	16x4	Tristate	3.49
74S287	1Kx1 Prom	7.95	91L02	1Kx1	Low Power	3.95
	SR'S		74200	256x1	Static	6.95
2401	2K DSR	9.95	74LS670	4x4	Register	3.95
2533	1K SSR	7.95	93410	256x1	Static	2.95

91L02 1K Static RAM Direct Replacement for 2102-1 with 40% Less Current Drawn. **2.49**

Vector General Purpose Logic CARD Board 14.95
*Very High Noise Immunity * Holds 12 ea. 14 pin DIPS
*44 pin Edge Connection

THE KILOBYTE RAM CARD Per Kit **$49.95**
*Complete 1Kx8 Memory * High Noise Immunity Components
*Single 5v supply *500NS Access Time & Kit includes sockets, ICS & Board



4 PDT SWITCHES, HIGH QUALITY P B. TYPES .69

KITS EXAR ICS

XR-2206KA SPECIAL $17.95
Includes monolithic function generator IC, PC board, and assembly instruction manual.

XR-2206KB SPECIAL $27.95
Same as XR-2206KA above and includes external components for PC board.

TIMERS

XR-555CP **SPECIAL**	$ .69
XR-320P	1.55
XR-556CP	1.85
XR-2556CP	3.20
XR-2240CP **SPECIAL**	3.25

PHASE LOCKED LOOPS

XR-210	5.20
XR-215	6.60
XR-567CP	1.95
XR-2567CP **SPECIAL**	2.99

STEREO DECODERS

XR-1310P	$3.20
XR-1310EP	3.20
XR-1800P	3.20

WAVEFORM GENERATORS

XR-205	8.40
XR-2206CP **SPECIAL**	4.49
XR-2207CP	3.85

MISCELLANEOUS

XR-2211CP	6.70
XR-2261	3.79

Special Requested Items

RC4194	Dual Track V Reg	$5.95	N8T97	$3.00	2533	$7.95
RC4195	±15V Track Reg	3.25	4024P	2.25	8263	5.95
F9368	Decoder	3.95	2513	11.00	8267	2.75
LD110/111	DVM Chip Set	28.00	2518	7.00	8288	1.15
CA3130	Super CMOS Op Amp	1.49	2524	3.50	8826	3.00
MC1408L7	A/D	9.95	2525	6.00	8880	1.35
F3341	FIFO	8.95	2527	5.00	7497	5.00

ZENERS—DIODES—RECTIFIERS

TYPE	VOLTS	W	PRICE	TYPE	VOLTS	W	PRICE
IN746	3.3	400m	4/1.00	IN4003	200 PIV	1 AMP	.10
IN751A	5.1	400m	4/1.00	IN4004	400 PIV	1 AMP	.10
IN752	5.6	400m	4/1.00	IN3600	50	200m	6/1.00
IN753	6.2	400m	4/1.00	IN4148	75	10m	15/1.00
IN754	6.8	400m	4/1.00	IN4154	35	10m	12/1.00
IN965B	15	400m	4/1.00	IN4734	5.6	1w	.28
IN5232	5.6	500m	.28	IN4735	6.2	1w	.28
IN5234	6.2	500m	.28	IN4736	6.8	1w	.28
IN5235	6.8	500m	.28	IN4738	8.2	1w	.28
IN5236	7.5	500m	.28	IN4742	12	1w	.28
IN456	25	40m	6/1.00	IN4744	15	1w	.28
IN458	150	7m	6/1.00	IN1183	50 PIV	35 AMP	1.60
IN485A	180	10m	5/1.00	IN1184	100 PIV	35 AMP	1.70
IN4001	50 PIV	1 AMP	.09	IN1186	200 PIV	35 AMP	1.80
IN4002	100 PIV	1 AMP	.10	IN1188	400 PIV	35 AMP	3.00

TRANSISTORS

MPS-A05	5/1.00			PN4249	4/$1.00
MPS-A06	5/$1.00	PN3567	3/$1.00	PN4250	4/$1.00
2N2219A	3/$1.00	PN3568	4/$1.00	2N4400	4/$1.00
2N2221	4/$1.00	PN3569	4/$1.00	2N4401	4/$1.00
2N2222A	5/$1.00	2N3704	5/$1.00	2N4402	4/$1.00
2N2369	5/$1.00	2N3705	5/$1.00	2N4403	4/$1.00
2N2369A	4/$1.00	2N3700	5/$1.00	2N4409	5/$1.00
FN2415	5/$1.00	2N3707	5/$1.00	2N5086	4/$1.00
2N2484	4/$1.00	2N3711	5/$1.00	2N5087	4/$1.00
2N2906A	4/$1.00	2N3724	$.65	2N5088	4/$1.00
2N2907A	5/$1.00	2N3725	$1.00	2N5089	4/$1.00
2N2925	5/$1.00	2N3903	5/$1.00	2N5129	5/$1.00
2N3053	2/$1.00	2N3904	4/$1.00	2N5138	5/$1.00
2N3055	$.89	2N3905	4/$1.00	2N5139	5/$1.00
MJE3055	$.89	2N3906	4/$1.00	2N5209	5/$1.00
2N3392	5/$1.00	2N4013	3/$1.00	2N5951	5/$1.00
2N3398	5/$1.00	2N4014	3/$1.00	C106B1 SCR	2/$1.00

CAPACITOR CORNER

50 VOLT CERAMIC DISC CAPACITORS

	1-9	10-49	50-100		1-9	10-49	50-100
10pf	.05	.04	.03	.001	.05	.04	.035
22 pf	.05	.04	.03	.0047	.05	.04	.035
47 pf	.05	.04	.03	.01	.05	.04	.035
100 pf	.05	.04	.03	.022	.06	.05	.04
220 pf	.05	.04	.03	.047	.06	.05	.04
470 pf	.05	.04	.035	.1	.12	.09	.075

100 VOLT MYLAR FILM CAPACITORS

.001mf	.12	.10	.07	.022mf	.13	.11	.08
.0022	.12	.10	.07	.047mf	.21	.17	.13
.0047mf	.12	.10	.07	.1mf	.27	.23	.17
.01mf	.12	.10	.07	.22mf	.33	.27	.22

± 20% DIPPED TANTALUMS (SOLID) CAPACITORS

.1	35V	.28	.23	.17	1.5	35V	.30	.26	.21
.15	35V	.28	.23	.17	2.2	25V	.31	.27	.22
.22	35V	.28	.23	.17	3.3	25V	.31	.27	.22
.33	35V	.28	.23	.17	4.7	25V	.32	.28	.23
.47	35V	.28	.23	.17	6.8	25V	.36	.31	.25
.68	35V	.28	.23	.17	10	25V	.40	.35	.29
1.0	35V	.28	.23	.17	15	25V	.63	.50	.40

MINIATURE ALUMINUM ELECTROLYTIC CAPACITORS

Axial Lead					Radial Lead				
.47	50	.15	.13	.10	.47	25	.15	.13	.10
1	50	.16	.14	.11	.47	50	.16	.14	.11
3.3	50	.15	.13	.10	1	16	.15	.13	.10
4.7	25	.16	.14	.12	1	25	.16	.14	.11
10	25	.15	.13	.10	1	50	.16	.14	.11
10	50	.16	.14	.12	4.7	16	.15	.13	.10
22	25	.17	.15	.12	4.7	25	.15	.13	.10
22	50	.24	.20	.18	4.7	50	.16	.14	.11
47	25	.19	.17	.15	10	16	.14	.12	.09
47	50	.25	.21	.19	10	25	.15	.13	.10
100	25	.24	.20	.18	10	50	.16	.14	.12
100	50	.35	.30	.28	47	50	.24	.21	.19
220	25	.32	.28	.25	100	16	.19	.15	.14
220	50	.45	.41	.38	100	25	.24	.20	.18
470	25	.33	.29	.27	100	50	.35	.30	.28
1000	16	.55	.50	.45	220	16	.23	.17	.16
2200	16	.70	.62	.55	470	25	.31	.28	.26

comparison can be made. For example, *Popular Electronics* came out with their HIT cassette interface, the *Computer Hobbyist* has another, the Southern California Computer Society has one and so does MITS. I understand that the Digital Group in Denver has a tape transport. Which is best?

Can I run MITS BASIC in any other mini? Does anyone make a mini I can run Datapoint software in?

I realize that this is quite an undertaking but think of the thousands of hours that will be saved if this information is available. You have made a good start with "Which Microprocessor for You?"

As you probably guessed I am an application man and over the years have developed quite a Fortran subroutine library. This makes it easy to write a program – just call, call, call! I hope to provide some useful articles later after I get my Altair up to being a workhorse instead of just a toy.

William D. Roch
Woodland Hills CA

ON LIFE AND LIFELINE

Dear Sirs:

The first issue of BYTE is excellent. BYTE looks as if it is going to be exactly what (in my opinion at least) a computer hobbyist magazine ought to be.

Just one slight correction; the inventor of the game of LIFE is John Horton Conway, a professor of mathematics at Cambridge University (England), not Charles Conway as reported. LIFE was first widely publicized in the October 1970 issue of *Scientific American* and according to Martin Gardner in the November 1971 issue of that periodical (page 121), a newsletter called LIFELINE was started in 1971 by Robert T. Wainright, 1280 Edcris Rd., Yorktown Heights NY 10598. However, I never subscribed to the newsletter and have no idea whether it is still going or not. Perhaps one of your other readers will be more informative . . .

Charles A. Dunning Jr.
Sterling Heights MI

1 AND 1 EQUALS 2?

BYTE:

An article in issue #2 prompts me to write about something: Have just started reading up on computers and associated equipment. One item I read was about AND gates and OR gates, and their logic. In my opinion they are designated incorrectly. Remember in grade school, when you had the arithmetic problem 1 + 1 = 2?

Most of us said "1 and 1 equals 2." In other words the + sign (plus sign) stood for *and*.

Now in Boolean logic they say that the + sign means OR. And the • means AND.

Would have been much better to have the + sign mean AND, and the • sign mean OR (the • sign looks more like the 0 or OR filled in than a plus sign), but it's too late to change this system, because it's been in use too long.

But maybe it's not too late to change the system used on page 43 of issue #2 of BYTE. This shows LED bit pattern: 0-LED on; ●-LED off. To me, the filled in zero looks more like an LED that is on, and it seems a better illustration would be: ●-LED on; 0-LED off.

The ASCII code for A would then look like this: ● 000 00● with the filled-in circles indicating the punched out holes.

BYTE looks great. Keep it up!

James C. Madsen
Fresno CA

The AND function is "logical product," the OR function is "logical sum." The symbols are of course merely the usual product and sum symbols of mathematics. The way it is said in grade school is colloquial but, unfortunately, not true.

THE SUNTRONIX MODEL KBD IV Keyboard is ideally suited as a general purpose ASCII Keyboard for data terminal applications. This keyboard more than meets the needs of the data entry market for long life and reliability.

The KBD IV utilizes the 2 key rollover solid state read-only MOS memory allowing encoded outputs to be strobed out as each key is depressed. A second key may be depressed concurrent with the first, but the second key encoded output will not be strobed out until the first key is released. This feature prevents ambiguity of character codes as a result of two keys being depressed in rapid succession.

ELECTRICAL SPECIFICATIONS:
* Voltage requirements – +5.0 V and –12.0 V
* Power consumption – less than 200 mw
* Outputs – standard ASCII; 8-bits + strobe
* Negative or positive logic output, jumper selectable
* Output connector – standard 14 pin DIP IC socket

MECHANICAL FEATURES:
* Size – 12¼" x 6¾" x 2½"
* High grade glass epoxy PC board
* Keyboard ROM SMC KR2376 40 pin MOS
* Electronic shift lock, not mechanical
* Keyswitches one integral assembly, not individual keys
* Switches have four-finger phosphor bronze contacts with gold inlay
* Keycaps are 2-shot high strength ABS plastic

These keyboards are available off the shelf in two forms – fully assembled and unconditionally warranteed against defects in manufacture or materials for a full ninety days, or in the more economical kit version. Kit parts are fully warranteed against defects in manufacture or materials for a full ninety days. In either version, full instructions are supplied for operation, including specifications and data sheets. In the kit version, complete instructions are supplied for the assembly process. A reasonably competent technician can completely assemble and test this keyboard in one evening. All parts needed are included.

INTRODUCTORY PRICES

Factory assembled – $59.95 ppd.
Complete kit, w/instructions – $54.95 ppd.
Please add $1.00 handling per order.
Minimum order – $5.00



SUNTRONIX company

6 KING RICHARD DRIVE, LONDONDERRY, N.H. 03053
603-434-4644



BOOK REVIEWS

The Bugbook III: Micro Computer Interfacing *by David G. Larsen, Peter R. Rony and Jonathan A. Titus. Published by E & L Instruments, Inc., 61 First St., Derby CT 06418; 1-203-735-8774. $14.95.*

This excellent book is highly recommended to all microcomputer experimenters, and is a "must" for 8080 users. It includes probably the best available tutorial discussion of hardware and software techniques for interfacing microprocessors to external devices, enabling them to act as device controllers or sequencers. It also includes an excellent introduction to programming concepts and the 8080 instruction set, and a very good discussion of interrupt handling.

Bugbook III is the fourth in a series of books prepared by the authors to teach the application of modern digital electronics to the control and monitoring of physical processes. *Bugbooks I* and *II* describe experiments with conventional TTL integrated circuits, while *Bugbook IIA* deals with asynchronous data communications using UARTs. Each of the *Bugbooks* describes experiments which are to be performed with the Micro Designer student-oriented interfacing system and the Mark 80 microcomputer, which are marketed by E & L Instruments, Inc. Authors Larsen and Rony are associated with Virginia Polytechnic Institute and State University, where they teach both regular and extension courses on digital electronics.

Bugbook III is organized into eight instructional units. Each unit begins with an introduction, a list of objectives, a glossary of definitions, and ends with a set of experiments, a test, and a review section which relates the material just covered to the initial list of objectives.

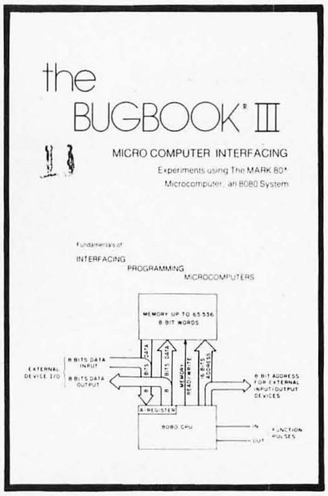
Unit 1 provides a general introduction and orientation to computers and microprocessors, discusses the basic tasks of interfacing a microprocessor to the outside world, and summarizes the characteristics of the 8080 microproccesor. Unit 2 describes breadboarding techniques specific to the Mark 80 microcomputer system. Unit 3 introduces the basic concepts of stored programs, instructions, and machine language, explains how instructions are decoded, and summarizes the 8080 instruction set in various ways.

Unit 4 describes ways of generating device select pulses to signal one of several input/output devices, and relates the 8080's input/output instructions to the device selection signals on the address lines. Unit 5 explains how microprocessor programs and timing loops can be used to provide time delays and sequencing for external control. Unit 6 is a lion-hearted attempt to clarify the complex details of decoding the 8080's external status signals in order to identify specific machine cycles which are entered as each instruction is executed.

Unit 7 further develops the techniques for input/output, and describes how the Mark 80 bus signals and control lines may be monitored (the Mark 80 is especially well designed for this purpose). Unit 8 discusses subroutines, interrupts, external flags and stacks, and includes a good introduction to the possible ways of dealing with interrupts.

The authors have done a really excellent job in several areas. One area is the definition of terms: Each unit begins with a glossary of definitions, and many definitions are repeated for emphasis in the context of the discussion in the text. Another area is the organization of the text for instructional purposes and the effort to provide motivation for the student. Still another area where the book succeeds is in the relation of hardware functions to software concepts.

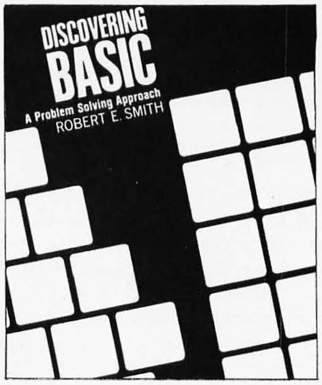
The material on the use of software timing loops and their use in sequencing and controlling external devices is difficult to find elsewhere. The discussion of interrupts covers the essential concepts of saving status information, enabling and disabling the interrupt system, and choosing among interrupts of different priorities.



The book does a good job of introducing basic programming concepts, and provides a large number of programming exercises which explore the 8080 instruction set in some depth. However, this Bugbook does not cover "higher-level" software concepts beyond the idea of simple loops and subroutines. All of the examples are presented in absolute machine language, and the use of assembly language is mentioned only briefly. There is no coverage of higher-level languages or of software "multi-tasking" approaches to the handling of multiple I/O devices and data streams. Perhaps these topics will be covered in another Bugbook.

This book fills an important gap in the currently available literature on microcomputers: It deals squarely with the interrelationships of hardware and software at the level of input/output device interfacing. This is in line with the authors' view of the role of microprocessors as device controllers rather than as general purpose computers. Readers who intend to connect their home computer system to their ham radio gear, model train system, electric range, burglar alarm system, or whatever, would do well to obtain this book. Any 8080 user who is a hardware or software hacker and is having trouble understanding the jargon of the "other side" would also profit from this book. The experiments depend to some degree on the layout of the Mark 80 system, but the text (with the exception of Unit 2) is applicable to all 8080 systems. For instructional purposes, this book is one of the best available on the market today. --d.h.f.

Bugbook, Micro Designer and Mark 80 are trademarks of E & L Instruments, Inc.



Discovering BASIC – A Problem Solving Approach *by Robert E. Smith, Hayden Publishing Co., New York NY, 1970. $7.95 hardcover, $5.95 paperback.*

Anyone who wants to learn BASIC for the Altair system or any other BASIC system should have this book. The book has very clear explanations that make it easy to comprehend.

The book is divided into 37 two- to three-page lessons. There are four review tests and four test correcting programs. Yes, you program the computer to correct your test for you. The second half of the book contains 50 review problems covering many different interests: algebra, geometry, finance, statistics, puzzles and games. At the end is a set of program solutions to problems in the book.

The reader will start to write simple programs after the first three pages. After that, more of the language is introduced bit by bit with activity reinforcement of each new command.

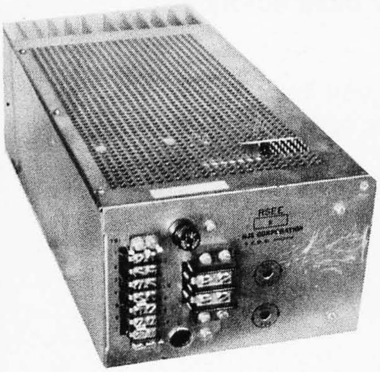
Lessons include: numbers and variables, order of calculations, built-in and user defined functions, flow charting, program looping, printing headings, inputs during execution, FOR-NEXT instructions, use of subroutines and matrices. All in all it is a very clear and comprehensive text for the learning of BASIC.

Thomas Charles Greer
224 N. Alabama St.
San Gabriel CA 91775

LOGIC POWER SUPPLIES

New Lambda transistorized, regulated, worth 2 or 3 times our price.

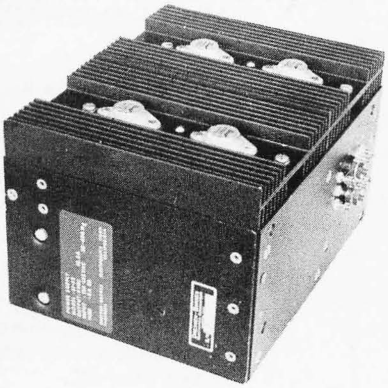
1. (LVEE – 5) OV, 5 VDC 74 amp 125.00
2. (LWD – 12) 12 VDC 26.5 amp 100.00
3. (LMB – 5) 5 VDC 3.7 amp 40.00
4. (LMD – 15Y) 15 VDC 9 amp 45.00



SOPHISTICATED LOGIC SUPPLIES

By Dressen-Barnes and NJE. Transistorized, finely filtered and regulated. 115 volt input.

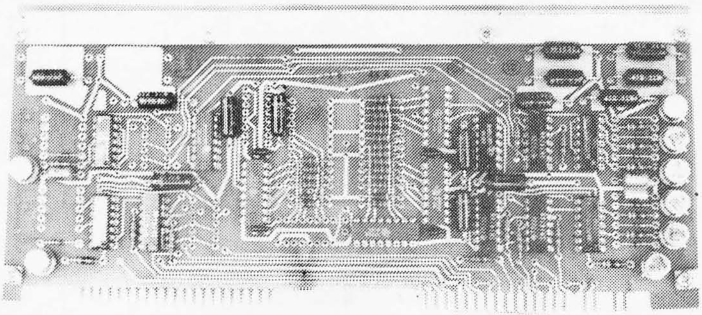
#61-5S	Output	5VDC 21 Amp 27 lb	$47.50
#51-5S	Output	5VDC 10.5 Amp 16 lb	35.00
#421-32	Output	32VDC 3.3 Amp 12 lb	20.00
#421-90	Output	90VDC 1.2 Amp 12 lb	20.00
#NJE		5VDC 34 Amp 35 lb	75.00



MEMORY SYSTEM $125.00

New memory system by Honeywell, small ... measures only 9x4x1 inches. 1024 core memory, 1024 words with 8,9,10 bits/word. Random access, with all logic, register, timing, control, core select and sense functions in one package. New, booklet of schematics and data. Looks like a good beginning for a mini-computer. Limited supply on hand.

Ship wgt 3 lbs. #SP-79 .... $125.00

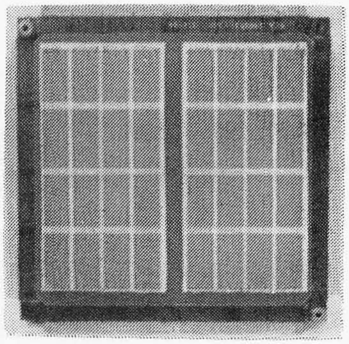


COMPUTER CAPS BRAND NEW						
	1,000 μF	15 volt	$ .35	2,000 μF	35 volt	1.00
	2,000	15	.50	19,500	25	$2.50
	1,000	25	.70	12,000	40	2.50
	3,000	25	1.00	3,900	50	2.00
	1,000	35	.80	22,000	75	3.25

CORE MEMORY

Another brand new memory, ultra small. Measures only 4 x 4 inches with format on one plane of 32 x 32 x 16 (16,384). Only about 35 units of this on hand.

#SP-81 .... $20.00



SANDERS 720 KEYBOARD $40.00

Meshna

FREE CATALOG

Please add shipping cost on above.

MESHNA PO Bx 62 E. Lynn Mass. 01904

BYTE reader service

To get further information on the products advertised in this issue of BYTE merely tear, rip, or snip out this advertiser index, fill out the data at the bottom of the page, mark the appropriate boxes, and send the works to BYTE, Peterborough NH 03458. Readers get extra Brownie Points for sending for information since this encourages advertisers to keep using BYTE – which in turn brings you a bigger BYTE.

ADVERTISER INDEX

- ☐ American Cancer Society 91
- ☐ ACM 15
- ☐ A. P. Products 2
- ☐ BYTE Subscriptions 95
- ☐ BYTE Retail Sales 110
- ☐ Bytronics 39
- ☐ Celdat 17
- ☐ Centi-Byte 18
- ☐ CMR 97
- ☐ Comp-Sultants 110
- ☐ Continental Specialties 81
- ☐ Delta Electronics 109
- ☐ Dutronics 98
- ☐ Godbout 6, 7
- ☐ Hickok 97, 99
- ☐ Iasis 56, 57
- ☐ IEEE 19
- ☐ Int'l Elec. Unltd. 103
- ☐ Intron 104
- ☐ James 105, 99
- ☐ Martin Research 16
- ☐ Meshna 111
- ☐ Micro Digital 77
- ☐ Mikra D 104
- ☐ MITS CIV, 40-47
- ☐ Ohio Scientific 102
- ☐ Processor Tech. 28, 29
- ☐ RGS 25
- ☐ Scelbi 62, 63
- ☐ Southwest Tech CIII
- ☐ Sphere 11, 13
- ☐ Suntronix 107
- ☐ Teleterminal 106
- ☐ Tri-Tek 101
- ☐ Visulex 102
- ☐ Windjammer 92, 93

Messages for the editor:

Reader's Service
BYTE
Green Publishing Inc.
Peterborough NH 03458

DECEMBER 1975

BYTE acquired via
☐ Subscription
☐ Newsstand
☐ Stolen

Please print or type.

Name ______________________

Address ______________________

City __________ State ______ Zip ______

Coupon expires in 60 days . . .

THE BYTE QUESTIONNAIRE

BYTE is dedicated to the needs of its readership. In order to better gauge matters of editorial policy and content, as well as to give our advertisers some "hard facts," we publish this questionnaire. In this month's list are a few more questions of editorial interest:

What applications do you have in mind for your own personal computer system?

Do you presently have a computer?

If you own or use a personal computer system, what is its CPU chip or main frame computer design?

Have you had any "on the job" training using computers as a tool for some purpose?

Do you know what a computer language is? Would you list the computer languages (if any) which you know?

These questions are a short form "letter to the editor." If you have additional comments, don't hesitate to write! Send completed questionnaires to BYTE, Dept. Q, Peterborough NH 03458.

Classified Ads Available for Individuals and Clubs

Readers who have equipment to buy, sell or swap should send in a clearly typed or printed notice to that effect. Insertions should be limited to approximately 100 words or less. Advertisements for this feature can be accepted from individuals or bona-fide clubs of computer users only. Commercial advertisers should contact Bill Edwards, BYTE advertising manager, for the latest rate card and terms. Individual/club classifieds will be printed free on a space-available basis in the earliest possible issue of BYTE.

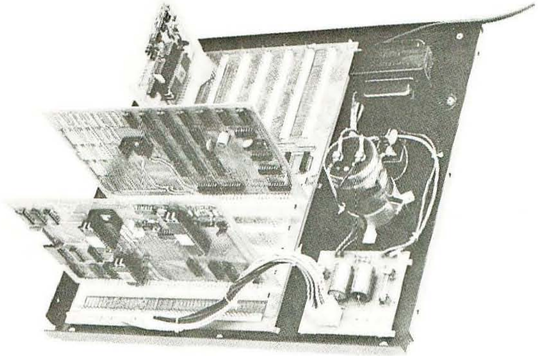
Feel free to photocopy this page if you wish to keep your BYTE intact.

SWTPC 6800

The Computer System You Have Been Waiting For

A BENCHMARK SYSTEM—Using the MOTOROLA M6800 benchmark microprocessor family.





Southwest Technical Products is proud to introduce the M6800 computer system. This system is based upon the Motorola MC6800 microprocessor unit (MPU) and it's matching family of support devices. The 6800 system was chosen for our computer because this set of parts is currently in our opinion the "Benchmark Family" for microprocessor systems. It makes it possible for us to provide you with a computer system having outstanding versitility and ease of use.

In addition to the outstanding hardware system, the Motorola 6800 has without question the most complete set of documentation yet made available for a microprocessor system. The 714 page Applications Manual for example contains material on programming techniques, system organization, input/output techniques, and more. Also available is the Programmers Manual which details the various types of software available for the system and provides instructions for the programming and use of the unique interface system that is part of the 6800 design. The M6800 system minimizes the number of required components and support parts, provides extremely simple interfacing to external devices and has outstanding documentation.

Our kit combines the MC6800 processor with the MIKBUG® read-only memory (ROM). This ROM contains the program necessary to automatically place not only a loader, but also a mini-operating system into the computers memory. This makes the computer very convenient to use because it is ready for you to enter data from the terminal keyboard the minute power is turned "ON". Our kit also provides a serial control interface to connect a terminal to the system. This is not an extra cost option as in some inexpensive computers. The system is controlled from any ASCII coded terminal that you may wish to use. Our CT-1024 video terminal is a good choice. The control interface will also work with any 20 Ma. Teletype using ASCII code, such as the ASR-33, or KSR-33. The main memory in our basic kit consists of 2,048 words (BYTES) of static memory. This eliminates the need for refresh interrupts and allows the system to operate at full speed at all times. Our basic kit is supplied with processor system, which includes the MIKBUG ROM, a 128 word static scratch pad RAM, and clock oscillator bit rate divider; main memory board with 2,048 words, a serial control interface, power supply, cabinet with cover and complete assembly and operation instructions which include test programs and the Motorola Programmers Manual.

If you have a Motorola 6800 chip set, we will sell you boards, or any major part of this system as a separate item. If you would like a full description and our price list, circle the reader service number or send the coupon today. Prices for a complete basic kit begin at only $450.00.

MAIL THIS COUPON TODAY

☐ Enclosed is $450.00 ☐ or Master C. #________ Bank #________

☐ or BAC #________ Ex Date________

For My SWTPC Computer Kit ☐ ☐ Send data package

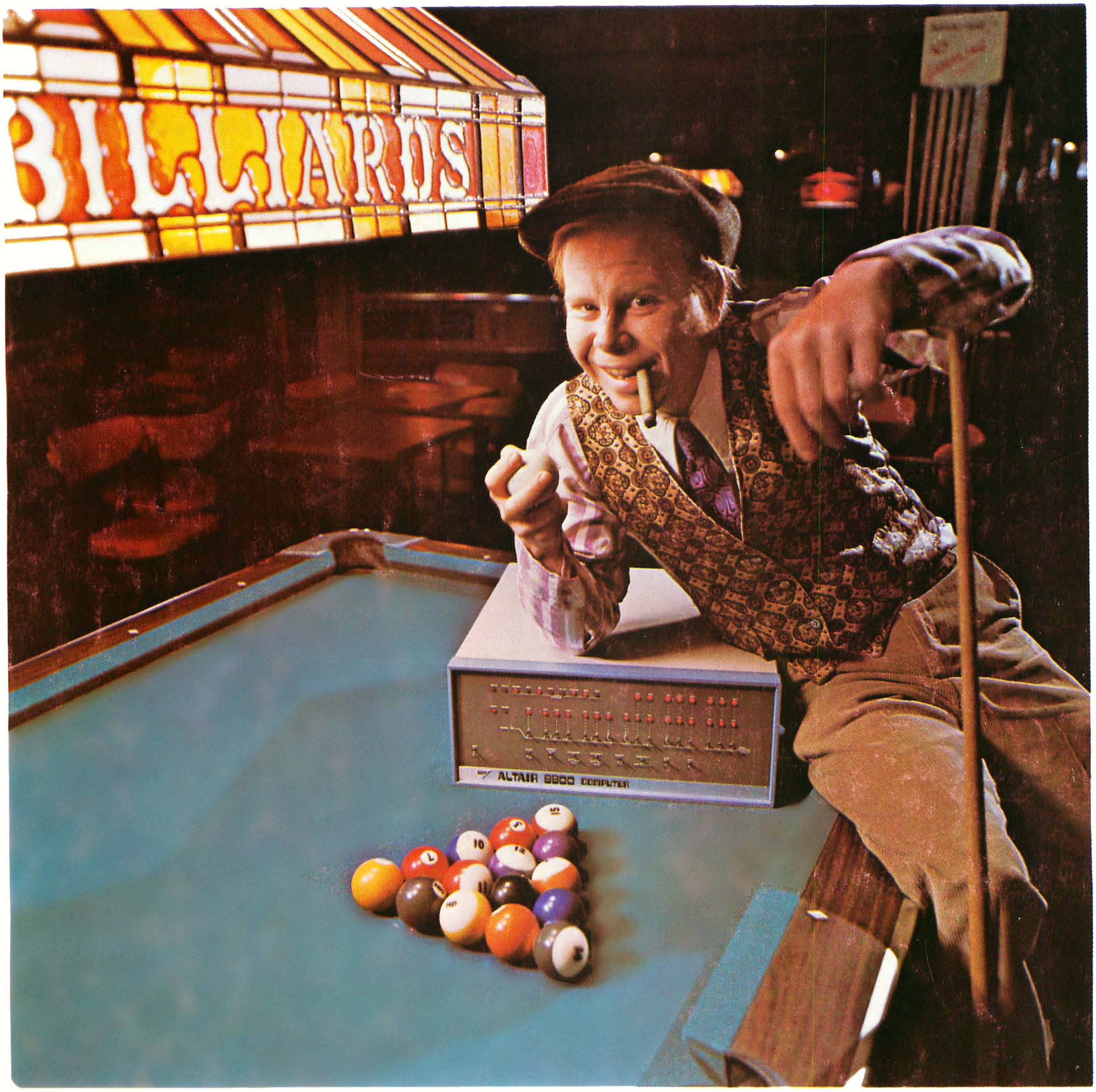
NAME ________

ADDRESS ________

CITY________ STATE ________ ZIP ________

Southwest Technical Products Corp., Box 32040, San Antonio, Texas 78284

The MITS Altair 8800.



(It's showing up in some of the most unusual places.)